M. SAAL

UTILITAIRES POUR ZX 81

POCHE *informatique* 9

Collection
POCHE INFORMATIQUE
dirigée par ALAIN TAILLIAR
directeur de la rédaction de la revue MICRO-SYSTEMES

UTILITAIRES
pour
ZX 81

ISBN 2-85535-070-0 ISSN 0757-6730

Marc SAAL

UTILITAIRES
pour
ZX 81

Diffusion :

ÉDITIONS TECHNIQUES ET SCIENTIFIQUES FRANÇAISES

2 à 12, rue de Bellevue, 75940 PARIS CEDEX 19

COLLECTION E.T.S.F. MICRO-SYSTEMES

1 — A. VILLARD et M. MIAUX, *Un microprocesseur pas à pas*
2 — A. VILLARD et M. MIAUX, *Systèmes à microprocesseur*
3 — P. GUEULLE, *Maîtrisez votre ZX 81*
4 — E. FLOEGEL, *Du Basic au Pascal*
5 — P. COURBIER, *Vous avez dit Basic ?*
6 — M. MARCHAND, *Vous avez dit micro ?*
7 — P. GUEULLE, *Pilotez votre ZX 81*
8 — M. JACQUELIN, *La micro-informatique et son A.B.C.*
9 — M. OURY, *Maîtrisez le TO 7*
10 — P. GUEULLE, *Pilotez votre ORIC*

COLLECTION POCHE-INFORMATIQUE

1 — G. ISABEL, *50 programmes pour ZX 81*
2 — P. GUEULLE, *Montages périphériques pour ZX 81*
3 — C. GALAIS, *Passeport pour Applesoft*
4 — R. BUSCH, *Passeport pour Basic*
5 — M. ROUSSELET, *Mathématiques sur ZX 81*
6 — C. GALAIS, *Passeport ZX 81*
7 — G. PROBST, *50 programmes pour CASIO FX-702 et 801 P*
8 — G. PROBST, *60 programmes pour CASIO PB 100*
9 — M. SAAL, *Utilitaires pour ZX 81*

Sommaire

Le but de cet ouvrage est essentiellement pratique. Il s'agit d'expliquer en détail la manière d'utiliser les ressources matérielles et logicielles intégrées dans le ZX 81.

Ces explications sont assorties d'applications sous la forme d'utilitaires en langage machine.

Les notions de base de la programmation en langage machine sont supposées acquises grâce à l'un des très nombreux ouvrages existant déjà.

Cependant le débutant pourra se contenter de rentrer les programmes donnés et de les utiliser sans en comprendre parfaitement le fonctionnement.

Les routines proposées sont listées très clairement grâce à un désassembleur dont la liste est fournie en annexe.

L'implémentation en machine pourra s'opérer soit à partir du Basic par l'un des nombreux procédés publiés dans la presse et la littérature, soit à l'aide d'un assembleur, soit enfin en employant le petit moniteur d'hexadécimal également fourni.

1

Principes de base

1/ L'utilisation de l'écran et du clavier.

Il est essentiel qu'un programme écrit en langage machine soit susceptible de communiquer avec l'utilisateur de la même manière qu'en Basic. C'est pourquoi la maîtrise de l'écran et du clavier constitue la première étape de l'apprentissage de l'utilisation des ressources de la machine.

A. L'écran

Rappelons brièvement comment l'écran est exploité.

Il existe, dans le ZX 81, une zone de la mémoire où sont conservés les codes représentant les caractères à afficher.

Lorsque l'image doit être produite (soit tous les 1/50e de seconde en mode SLOW), le microprocesseur est interrompu dans la tâche qu'il accomplit et va adresser la série des codes.

Or, par une astuce de construction, le Z 80 est rendu incapable de lire effectivement ces codes. En effet, le circuit-maître du ZX 81 le place en mode attente puis lui passe un code 00 (NOP), ce qui signifie que le microprocesseur ne fait rien (ou presque).

Parallèlement, le circuit maître utilise les données adressées pour réaliser l'affichage.

La fin d'une ligne d'écran est représentée en mémoire par le code 76_h (lire sept six hexa), celui de New-Line. Or, 76_h correspond à l'instruction HALT du microprocesseur. C'est la seule que le circuit-maître daigne transmettre au Z 80.

Ce dernier ne fait rien de plus mais ne lit plus la mémoire jusqu'à ce qu'une nouvelle interruption survienne.

Celle-ci renvoie à un sous-programme qui décompte les lignes. Quand l'écran complet (24 lignes) a été rafraîchi, le microprocesseur reprend sa tâche interrompue.

• Écriture d'un caractère sur l'écran

La manière la plus intuitive d'inscrire un caractère consiste à placer le code de celui-ci à l'emplacement voulu dans l'image de l'écran en mémoire.

Nous savons que cette image se trouve à partir d'une adresse conservée dans la variable système D-FILE en 16396-16397 (ou $400C_h$-$400D_h$).

Il suffirait donc de faire par exemple :

```
LD HL, (400C)
INC HL
LD A, 3D        (3D est le code de « X »)
LD (HL), A
RET
```

pour afficher la lettre X au début de l'écran.

Cela marche bien et on peut penser remplir l'écran de « X » en réalisant une boucle incrémentant HL. Mais dans ce cas, la machine « se plante ».

Que s'est-il passé ?

En fait, la boucle dont nous parlions va installer le code de « X » ($3D_h$) partout dans la mémoire d'écran sans respecter le fameux code 76_h (fin de ligne). Et voilà le système privé de ses points de repère, égaré !

Retenons donc que :

Si on écrit directement dans la mémoire d'écran, il faut respecter les codes de fin de ligne.

Cette façon de gérer l'écran est à choisir essentiellement pour les jeux graphiques où des objets se déplacent sur l'écran.

On a besoin dans ce cas de savoir où se trouve l'objet et il s'avère pratique de se représenter sa position par une simple adresse.

Pour notre part, nous préférons : **RST 10** (lire Restart un zéro).

Les « Restart » représentent des instructions spéciales des microprocesseurs de la famille du Z 80. Elles permettent d'appeler en un seul octet (en l'occurence D7 pour RST 10) un sous-programme. On les réserve à des routines très fréquemment employées (pour économiser 2 octets à chaque appel).

Par exemple, dans le ZX 81 :

— **RST 08** vide la pile des retours de sous-programmes et affiche un code d'erreur. Ainsi :
 RST 08
 OC
 sort sur un compte rendu D (soit $OC_h + 1$).

— **RST 18** et **RST 20** lisent les caractères que l'interpréteur Basic devra exploiter.

— **RST 28** permet d'accéder au calculateur.

RST 10 affiche à la position courante de l'écran le caractère dont le code est contenu dans l'accumulateur A.

A peut avoir les valeurs suivantes :
— de 00 à $3F_h$ inclus
— de 80_h à BF_h inclus
— 76_h (New Line).

Tout autre code fait perdre le contrôle du système.

RST 10 présente l'avantage majeur de ne pas modifier les principaux registres du Z 80 et de respecter les signaux de fin de ligne. De plus, si l'écran se remplit, un retour au Basic s'effectue sur le compte rendu 5. Cela peut constituer un moyen de quitter une routine en langage machine en dehors de l'habituel RET.

Ainsi, remplir l'écran de « X » devient enfantin :

```
Début    LD A, 3D
         RST 10
         JR Début
```
(sortie sur le compte rendu 5).

Routine CLS

En faisant : **CALL 0A2A,** on efface l'écran.

Routine SCROLL

SCROLL s'obtient par **CALL OECE.**

Routines PRINT AT

On peut choisir la prochaine position d'écriture de deux manières :

1) LD BC, 0105
CALL 08F5

équivaut à PRINT AT 1,5 ;

2) LD BC, 1821
CALL 0918

équivaut à PRINT AT 0,0 ;

Dans ce cas, B prend les valeurs 18 $_h$ à 2 de haut en bas de l'écran et C vaut 21 $_h$ à 1 de gauche à droite de chaque ligne.

Dans les deux cas, HL contient au retour l'adresse de la case mémoire image de l'emplacement d'écran choisi.

La seconde méthode présente l'avantage suivant : les valeurs de B et de C sont conservées dans la variable-système S-POSN (4039 $_h$-403A $_h$). Ainsi, on peut en lisant ces cases de mémoire savoir à tout moment quelle est la position d'écriture. Il est alors possible de modifier cette dernière dans le sens voulu (par exemple : monter d'une ligne grâce à :

```
LD BC, (4039)
INC B
CALL 0918).
```

B. Le clavier

La scrutation du clavier s'effectue de différentes façons :

CALL 02BB : permet de récupérer dans HL un code spécifique de la touche pressée. Même si la touche n'a pas été relachée, elle sera prise en compte à chaque appel de ce sous programme. Cette routine ne s'avère pas très pratique car il faut disposer en permanence de la table des codes ramenés dans HL. En mode SLOW, tout se simplifie.

En effet, dans ce mode, la pression d'une touche provoque la levée d'un drapeau (un signal en quelque sorte) grâce à une interruption. Ce drapeau est l'élément binaire (le bit) le plus bas (d'ordre 0) de l'octet CDFLAG (adresse : $403B_h$).

Attendre une touche revient alors à patienter jusqu'à ce qu'une interruption vienne lever ce signal. Notons que le microprocesseur ne fera rien d'autre pendant ce temps (à part des tâches de gestion : rafraîchissement de l'écran et de la mémoire).

Une fois la touche pressée, son code spécifique est récupérable dans la variable-système LAST-K (adresse : 4025_h-4026_h) et on peut en déduire le code ordinaire grâce à la routine d'adresse $07BD_h$. N'oublions pas qu'il faudra aussi rabaisser le drapeau (grâce à CALL OF4B).

Finalement, en mode SLOW, la routine suivante :

```
Start   BIT 0, (IY + 3B)     IY contient 4000 h
        JR Z, Start
        LD BC, (4025)
        CALL OF4B
        CALL 07BD
        LD A, (HL)
        RET
```

attendra une touche et ramènera son code dans A.

• Test des caractères

On peut utiliser certains tests intégrés à la ROM :

CALL 14D2 ramène le carry (retenue) à 1 si et seulement si le code contenu dans l'accumulateur correspond à une lettre ou un chiffre.

CALL 14CE distingue exclusivement les lettres (de la même manière).

2/ Les routines utiles

Ne nous attardons pas sur :
CALL 0F23 qui passe en mode FAST
et CALL 0F2B qui ramène en mode SLOW.

On doit cependant rappeler que certaines fonctions du ZX 81 ne fonctionnent qu'en FAST. Pourtant, il est possible de travailler en permanence en mode SLOW.

Le secret est que le drapeau qui signale le mode en cours possède un double. Ainsi, lorsque la machine a besoin passagèrement de FAST, elle appelle la routine d'adresse $02E7_h$: CALL 2E7. 743

Celle-ci modifie le double mais laisse intact le vrai drapeau.

Le travail effectué, CALL 207 permet de rétablir les conditions initiales. 519

Nous verrons l'importance de tout cela un peu plus loin.

A. Insertions/suppressions

Le ZX 81 possède un aspect sans doute unique en micro-informatique. En effet, la répartition de la mémoire n'est pas fixe. Dans le but d'économiser de la place, chaque zone se trouve réduite au minimum.

La machine se souvient des limites de chacune grâce à une série de pointeurs : les variables-système.

Chaque fois qu'on insère ou qu'on supprime un octet, tous les pointeurs doivent être réactualisés sinon le système s'égare et on en perd le contrôle.

De ce fait, il existe des routines spécialisées dans la ROM. Elles se chargent de déplacer les contenus des blocs de mémoire en corrigeant les pointeurs au besoin.

• Insertions

CALL 099B ouvre un espace dans la liste Basic, l'écran ou la zone des variables, à l'emplacement pointé par HL.

Par exemple : LD HL, 4082
CALL 099B
RET

déplacera le contenu de la case mémoire d'adresse 4082 $_h$ et le disposera en 4083 $_h$. Celui de 4083 $_h$ se retrouvera en 4084 $_h$ et ainsi de suite. On aura ainsi libéré l'emplacement d'adresse 4082 $_h$.

Au retour, HL sera diminué de 1 (4081 $_h$ dans l'exemple), DE pointera sur la case libérée (4082 $_h$).

CALL 99E ouvre le nombre d'espaces correspondant à la valeur du registre pair BC. Par exemple :

```
LD BC, 0008
LD HL, 4082
CALL 099E
RET
```

libérera 8 espaces à partir de 4082 $_h$ (16514). HL et DE subissent le même sort que précédemment.

On peut se dire que cela permettrait de résoudre un problème habituel sur le ZX 81, à savoir la création de REM longues pour y loger du langage machine.

Cette possibilité existe en effet, à condition de se souvenir que la ligne contenant la REM se trouve allongée de ce fait. Si l'on n'y prend garde, la perte de contrôle est certaine. Il conviendra donc de corriger les octets conservant la longueur de la ligne en les augmentant du nombre des emplacements ouverts. Mais oublions un instant ce détail.

Essayons donc à présent le programme suivant :

```
LD BC, 00A0
LD HL, 4082
JP 099E
```

Note : JP 99E équivaut à CALL 99E suivi de RET.

Nous pensons obtenir ainsi 160 (=AO $_h$) espaces libres à la suite d'une REM que nous aurons préalablement disposée en ligne 1.

En fait, 9 fois sur dix, le système « se plante ».

Que se passe-t-il ?

Au cours des déplacements des contenus des blocs de mémoire, si l'on se trouve en mode SLOW, une interruption est susceptible de survenir. Et, si la mémoire image de l'écran n'a pas encore fini d'être complètement transférée, le système peut se trouver privé de ses points de repère.

Cela n'arrive pratiquement jamais quand on ouvre un seul espace (CALL 099B) mais demeure très fréquent quand on emploie CALL 099E.

La solution consiste à passer en mode FAST ainsi que nous l'avons expliqué plus haut :

```
LD BC, 00A0
LD HL, 4082
CALL 02E7
CALL 099E
JP 0207
```

fonctionne sans problème.

Nous verrons, en application, comment créer une REM de la longueur désirée.

Note : certains préfèrent utiliser CALL 0526 à la place de CALL 099B. La seule différence est que, dans ce cas, le contenu de l'emplacement libéré n'est pas laissé intact mais remplacé par la valeur de l'accumulateur A.

• Suppressions

Comme pour les insertions, il existe des routines spécialisées dans la ROM, chargées de « supprimer » des octets devenus inutiles. En fait, cela consiste à copier dans ces cases de mémoire les contenus des cases d'adresses supérieures et à corriger les variables-système.

CALL 0A60 « descend » les contenus des emplacements de mémoire à partir de celui pointé par HL et jusqu'à l'adresse conservée dans la variable-système STKEND. La valeur stockée dans la paire de registres BC indique le nombre d'octets à surcharger. Comme pour les insertions, il est nécessaire de passer en mode FAST si BC vaut plus que 1.

Exemple :
```
LD HL, 4082
LD BC, 0003
CALL 02E7
CALL 0A60
JP 0207
```
transférera le contenu de 4085_h en 4082_h (3 emplacements), celui de 4086_h en 4083_h, celui de 4087_h en 4084_h, etc. Ce qui se trouvait dans les 3 premières cases de mémoire (4082_h-4083_h-4084_h) est perdu.

Fréquemment, le problème consiste plutôt à effacer ce qui se trouve entre deux adresses (voir par exemple le programme d'élimination simultanée de plusieurs lignes du Basic, donné plus loin).

Dans ce cas, il faut placer :
— l'adresse du début du bloc à surcharger dans DE,
— l'adresse du début du bloc à transférer dans HL.

DE devra rester inférieur à HL. BC peut être quelconque.

On fait ensuite :
CALL 0A5D au lieu de CALL 0A60.

La différence (HL) — (DE) sera ainsi évaluée et transférée dans BC.

Exemple : transférer le contenu du bloc de mémoire commençant en 7000_h dans la zone de mémoire partant de 6000_h :
```
LD HL, 7000
LD DE, 6000
CALL 02E7
CALL 0A5D
JP 0207
```

Note : Pour surcharger *un seul* emplacement (celui que pointe HL), on peut simplement appliquer : CALL 055C.

B. Protection du haut de la mémoire

Chacun connaît la manière classique de protéger une portion de la partie haute de la mémoire grâce à la modification de la variable-système RAMTOP suivie d'un NEW.

Cependant, il arrive qu'on oublie de procéder à ces manœuvres, préalablement à l'entrée d'un programme, depuis le clavier ou la cassette.

On doit alors recommencer cette entrée puisque NEW provoque la perte de la liste et des variables.

Voici une solution à ce problème.

Voyons d'abord quel est le travail de la routine NEW :

1) placer en haut de la mémoire (adresse pointée par RAMTOP) la valeur $3E_h$ signalant la fin de la liste des numéros de lignes où renvoient les RETURN du Basic ;

2) initialiser le pointeur de pile de la machine (SP) ;

3) disposer la valeur que devra prendre SP en cas d'erreur dans la variable système ERR-SP ;

4) procéder encore à une série d'initialisations (registres I et IY, drapeaux, écran, etc...), supprimer le programme Basic et ses variables.

Nous n'avons pas besoin de cette dernière étape, mais nous pouvons imiter les autres grâce à la petite routine suivante :

```
2A-04-40   LD HL, (RAMTOP)    }
2B         DEC HL             }   Étape 1
36-3E      LD (HL), 3E        }
2B         DEC HL             }
F9         LD SP, HL          }   Étape 2
2B         DEC HL             }
2B         DEC HL             }
22-02-40   LD (ERR-SP), HL    }   Étape 3
C3-13-04   JP 0413            }
```

Ces quelques instructions peuvent être disposées en n'importe quel endroit de la mémoire.

Supposons qu'on les implante à partir de 16514.

Pour protéger le haut de la mémoire à partir de l'adresse 24576 (6000_h), on fera :
POKE 16388,0
POKE 16389,96 ($96 = 60_h$)
pour préparer RAMTOP.

Ensuite, RAND USR 16514 réalisera la protection sans altérer ni programme, ni variables. Au retour de cette routine, la liste Basic s'inscrira sur l'écran.

C. Sortie d'un programme en langage machine

Normalement, on quitte une routine en langage machine par un RET qui renvoie au Basic.

Nous avons vu plus haut qu'une sortie sur le compte rendu 5 (écran plein) est également possible si on emploie RST 10.

Un problème se pose lorsqu'un programme en assembleur utilise lui-même des sub-routines. Dans ce cas, il arrive qu'un certain nombre d'adresses de retours de sous-programmes se trouvent disposées sur la pile de la machine et un simple RET ne parviendra pas à relancer directement le Basic.

Parfois, il sera possible de vider la pile par des instructions POP (voir le programme de renumérotation plus loin).

Mais on aura plutôt intérêt à employer RST 08 qui réinitialise le registre SP.

RST 08 doit être suivi d'un octet dont la valeur x peut varier de 00 à 34 (22_h). Le compte rendu d'erreur affiché sera alors la lettre ou le chiffre dont le code vaut : $x + 29$ ($1D_h$).

Par exemple : RST 08
21_h
donnera : Y/O car le code de « Y » est $3E_h$ ($= 21_h + 1D_h$).

Pour x compris entre 00 et $0E_h$, le calcul se simplifie : il suffit d'augmenter l'octet en question de 1 en hexadécimal.

Par exemple : $0B_h$ donne une erreur C (= $0B_h + 1$).

Durant l'exécution d'une routine en langage machine, la touche BREAK n'est plus testée. Pour revenir au Basic à volonté, il conviendra donc d'inclure soi-même un test dans cette routine.

Si le clavier est scruté régulièrement, on pourra par exemple détecter (grâce à une instruction de comparaison) une touche précise et sortir alors par l'un des procédés indiqués (voir pour exemple le moniteur d'hexadécimal donné en deuxième partie).

Sinon, un appel à la routine d'adresse $0F46_h$ ramènera la retenue (carry) à 0 si la touche BREAK est pressée (le carry se teste facilement par RET C, RET NC, JR C etc.).

En fait, cette touche est connectée à l'élément binaire le plus bas (bit 0) du port d'entrée/sortie FE_h. Il s'avère donc possible de l'examiner directement :

```
LD A, 7F      présume que BREAK n'est pas activée
IN A, (FE)    lit le port FE h
RRA           sort le bit de droite (bit 0)
```

La routine en $0F46_h$ fait exactement cela. Nous verrons une application de cette notion dans la manipulation du magnétophone en tant que périphérique.

3/ Le calculateur

A. Représentation des nombres

La représentation des nombres dans la mémoire de la machine est dite « en virgule flottante » (en anglais « floating point »).

On peut comparer cette écriture à celle que l'on nomme « notation scientifique », dans le système décimal. Dans cette notation, 3000 s'écrira 3×10^3 et 2530 deviendra $2{,}53 \times 10^3$ ou $0{,}253 \times 10^4$.

Mais, contrairement à ce que laisserait penser la dénomination « virgule flottante », la place de la virgule dans ce procédé doit être fixe. Par exemple, on décidera de la disposer après le premier chiffre non nul du nombre.

Ainsi les formes suivantes resteraient valables :
3×10^3 $2{,}53 \times 10^3$ $4{,}7 \times 10^6$.

Celles-ci seraient rejetées :
30×10^2 $0{,}253 \times 10^4$ 47×10^5.

Dans l'expression $4{,}7 \times 10^6$, on distingue :
— 10 : la base
— 6 : l'exposant
— 4,7 : la mantisse.

Les concepteurs du ZX 81 ont organisé la représentation des nombres de la manière suivante :

— la base est nécessairement 2 car le microprocesseur travaille en binaire (cette base est implicite : on ne la rappelle nulle part) ;

— la mantisse doit être comprise entre 1/2 inclus et 1 exclu. Notons que 1/2 s'écrit en binaire 0,1 (c'est-à-dire 1 multiplié par 2 à la puissance moins 1, soit $1 \times 1/2$).

Comme il s'agit de la valeur minimum de la mantisse, on comprendra que cette dernière commencera toujours par 0,1 suivi d'une série de 1 et de 0. La place fixe de la virgule évoquée ci-dessus se situe dans ce cas devant le 1 binaire de rang le plus haut.

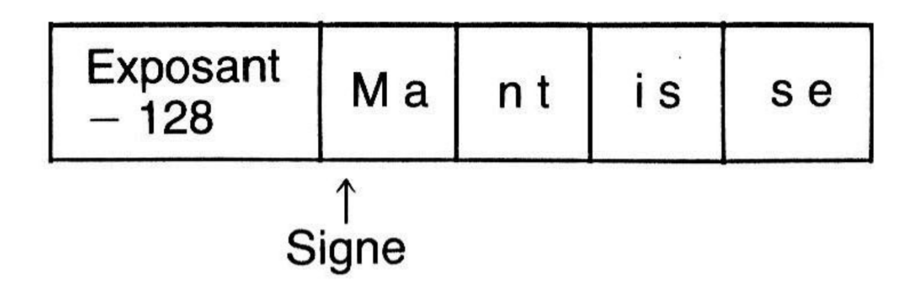
On constate alors que le 1 en question n'offre plus aucun intérêt. Pourquoi le conserver en mémoire alors qu'on connaît de toute façon son existence juste après la virgule ?

Voilà pourquoi, dans le ZX 81, le premier élément binaire de la mantisse est récupéré pour indiquer le signe de celle-ci. Il vaudra 1 pour un nombre négatif et 0 pour un nombre positif.

Rappelons enfin que la mantisse occupe 4 octets (soit 32 éléments binaires), ce qui limite la précision de la représentation.

— l'exposant occupe un seul octet. Sa valeur est diminuée de 128 (80_h). Cette réduction permet d'accroître la plage des valeurs susceptibles d'être représentées par ce système. Notons que, à une retenue près, soustraire ou ajouter 128 donne le même résultat, ce qui simplifiera nos calculs.

En résumé, un nombre occupe 5 octets selon le schéma suivant :



Exemples :

1) Représentation de 447 décimal :
447 décimal = 1 1011 1111 binaire.

Ramenons la mantisse dans la plage 1/2 à 1 :
$= 0{,}1101\ 1111\ 1000 \times 2^9$.
(Comme en décimal, on peut ajouter des zéros non significatifs après la virgule. On s'arrange pour que le nombre

total de chiffres suivant la virgule soit un multiple de 4, ce qui permet un passage aisé à l'hexadécimal).

Octet d'exposant : 9 + 128 = 137 (= 89_h)

Octets de mantisse : la mantisse vaut 1101 1111 1000
447 étant positif, on annule le premier 1 :
0101 1111 1000
soit en hexadécimal 5 F 8
Représentation de 447 : 89 5F 80 00 00.

2) Représentation de 5,375 décimal :
5,375 = 101,011 binaire
= 0,1010 1100 × 2^3
Exposant : 3 + 128 = 131 (= 83_h)
Mantisse : 0010 1100 (nombre positif)
Hexa : 2 C
Représentation : 83 2C 00 00 00.

3) Représentation de 0,285 :
0,285 = 0,01001...

Le développement binaire de ce nombre est illimité. Sa représentation ne tiendra compte que des 32 premiers éléments binaires. Cette troncature explique l'incapacité où se trouve la machine à mémoriser tous les nombres (même ceux qui paraissent simples en décimal) avec une précision parfaite.

4) Représentation de 0 :
Zéro bénéficie d'une représentation spéciale : 00 00 00 00 00.

Les calculs sur les nombres en virgule flottante se résument à des additions/soustractions et à des décalages. Un microprocesseur exécutera aisément ce type d'opérations.

Afin d'harmoniser les codages, les chaînes de caractères sont également conservées sur 5 octets. Le premier indique s'il s'agit d'un élément de tableau de chaînes, les 2 suivants contiennent la longueur, les 2 derniers pointent sur l'adresse où se trouve effectivement le premier caractère de la chaîne.

B. La pile du calculateur

Il convient de bien distinguer cette pile de celle de la machine.

— Le microprocesseur pousse sur la pile de la machine des adresses de retours de sous-programmes ou bien des registres pairs (instructions PUSH). Les transferts se font donc par paires d'octets. La progression de la pile machine s'opère vers les adresses basses, c'est-à-dire qu'au fur et à mesure des empilements/dépilements la mémoire est remplie de haut en bas et vidée de bas en haut.

Le registre SP du Z 80 pointe en permanence sur le prochain emplacement à garnir.

— La pile du calculateur progresse en sens inverse. Elle débute à une adresse conservée dans la variable système STKBOT et s'achève en STKEND. Les valeurs poussées sur cette pile correspondent à des nombres ou à des chaînes de caractères. Les transferts concernent donc 5 octets à chaque fois.

C. Le calculateur proprement dit

Maintenant que nous connaissons les structures qui le servent, nous pouvons examiner le calculateur proprement dit.

Il s'agit d'un ensemble de sous-programmes intégrés à la ROM et qui réalisent des opérations sur les valeurs disposées sur la pile du calculateur.

Il existe 4 types d'opérations :

1) Les transferts consistent à placer sur la pile ce qui se trouve dans d'autres emplacements de la mémoire ou dans des registres du microprocesseur et vice versa. La hauteur de la pile (différence STKEND-STKBOT) se trouve donc modifiée en plus ou en moins.

2) Les opérations unaires, c'est-à-dire qui n'utilisent qu'un seul argument comme par exemple SIN, COS, ABS, etc. Dans ce cas, la valeur qui se trouve sur « le dessus » de la

pile (la dernière empilée) est prise comme argument. Après modification, le résultat vient remplacer cette valeur qui est perdue de la sorte. Il n'y a pas d'altération de la hauteur totale de la pile.

3) **Les opérations de composition** consistent à combiner deux valeurs selon une loi afin d'élaborer un résultat (par exemple : +, —, ×, /).

Dans ce cas, le nombre placé sur le dessus de la pile sert d'opérateur, celui qui se situe juste en dessous représentant l'opérande. Le résultat remplace l'opérande que l'on perd donc de même que l'opérateur car la pile diminue de hauteur.

Prenons l'exemple d'une division :

STKEND →	A	→ / →	STKEND →	
STKBOT →	B		STKBOT →	B/A

Les utilisateurs de calculatrices de poche à notation polonaise inversée et pile opérationnelle connaissent bien ces notions. Nous verrons plus loin d'autres analogies avec ces machines.

4) **Les utilitaires** comprennent un ensemble d'opérations annexes comme la duplication, les tests, les sauts, etc. Ils demeurent indispensables et affectent parfois le contenu et la hauteur de la pile.

D. Utilisation du calculateur

Certains sous-programmes du calculateur sont appelés par le classique CALL. Mais les concepteurs du ZX 81 ont simplifié l'emploi de la plupart des routines grâce au système suivant :

— l'entrée dans le calculateur s'effectue par RST 28 ;

— à la suite de cette instruction, on dispose une série d'octets indiquant les opérations à effectuer sur le contenu de la pile.

Ainsi on gagne 2 octets à chaque appel de sous-programme. Par exemple, pour additionner les 2 nombres situés au sommet de la pile, on placera OF_h (au lieu de CALL 1755) après RST 28.

D'autre part, cette représentation s'avère plus parlante à l'esprit. En effet, on pourra donner un nom (pseudo-mnémonique) à l'opération associée à chaque octet. Par exemple, ADD équivaudra à OF_h. Ainsi une série de calculs complexes sera susceptible de s'écrire sous la forme d'une liste de pseudo-mnémoniques très lisibles.

Cela n'est pas sans rappeler les macro-instructions bien connues des habitués du langage assembleur ;

— enfin, l'octet 34_h signale la fin de l'utilisation du calculateur. Sa présence s'avère tout à fait indispensable car, après RST 28, aucune instruction du Z 80 n'est exécutable. Tout octet appellera une opération jusqu'à la rencontre de 34_h (FIN).

1) Les transferts vers la pile

CALL 1520 et **CALL 151D** placent respectivement les contenus de BC et de A sur la pile dont la hauteur s'accroît (de 5 octets). Cela permet d'empiler très facilement des nombres compris entre 0 et 65535. Par exemple :

```
LD BC, 00A0
CALL 1520
```

empilera la valeur $A0_h$ (= 160) convertie en virgule flottante.

D'autre part, il existe des instructions spéciales du calculateur pour certaines valeurs : (à la suite de RST 28)

$A0_h$ (EMP-0)	empilera	zéro
$A1_h$ (EMP-1)	»	un
$A2_h$ (EMP-1/2)	»	1/2
$A3_h$ (EMP-PI/2)	»	PI/2
$A4_h$ (EMP-10)	»	dix.

Pour tous les autres nombres, plusieurs cas se présentent :

a) Si un nombre en virgule flottante (codage Sinclair) se trouve à un endroit connu de la mémoire, on peut le pousser sur la pile de façon simple.

Il suffit de réserver 5 octets au sommet de la pile en y disposant un zéro (par exemple) puis d'employer une instruction de transfert automatique LDIR.

Cela donne :

LD HL, (Ad)	HL pointe sur l'adresse où se situe le nombre représenté en virgule flottante.
PUSH HL	
RST 28	Appel au calculateur.
EMP-0	Empile 0 (Octet A0 $_h$).
FIN	Quitte le calculateur (octet 34 $_h$).
EX DE, HL	Au retour, HL pointe sur l'adresse de la première des 5 cases mémoire libérées. On passe cette adresse dans DE.
POP HL	HL récupère Ad.
LD BC, 0005	5 octets à transférer.
LDIR	Instruction automatique.
RET	

Nous verrons une application de cette technique dans la routine de listage des variables Basic proposée en seconde partie.

b) Si l'on connaît le codage en virgule flottante d'un nombre, il est possible de l'empiler directement grâce à l'instruction EMP-DONNEE (octet 30 $_h$) du calculateur.

Cependant, la procédure demeure assez complexe et nous ne la recommandons pas. Voici néanmoins l'explication de son fonctionnement.

Après 30 $_h$ annonçant qu'une donnée à empiler va suivre, on trouvera :

D'une part, une première valeur servant à déterminer le nombre total d'octets qui représenteront la mantisse de cette donnée. Ce nombre est codé par les deux éléments binaires les plus significatifs de la première valeur.

Par exemple :

Première valeur	B7 $_h$
Binaire	[10] 11 0111
2 bits de poids fort	2
Il faut ajouter 1	2 + 1 = 3

Donc, 3 octets représenteront la mantisse.

Les éléments restant de la même première valeur (6 bits de poids faible) permettent la détermination de l'exposant.

Si ces 6 bits ne sont pas nuls, 50 $_h$ leur est ajouté pour donner l'exposant.

S'ils s'avèrent nuls, la première valeur sera négligée et l'octet venant après sera pris comme exposant après ajout de 50 $_h$.

Exemple :

Reprenons	B7
Binaire	10 [11] [0111]
Soit	3 7 = 37 $_h$ non nul
Exposant	37 $_h$ + 50 $_h$ = 87 $_h$.

Autre exemple :

Première valeur	C0 $_h$ suivie de 35 $_h$	
Binaire	[11] 00 0000	
Poids fort	3	donc 4 octets de mantisse.
Poids faible	0 0	nul
Exposant	35 $_h$ + 50 $_h$ = 85 $_h$	

(on prend l'octet suivant).

Notons que, dans ce dernier cas, l'octet représentant l'exposant ne sera pas décompté parmi les 4 à prendre. Il viendra en surcroît.

D'autre part, la mantisse. Comme cette dernière exige de toute façon 4 octets, ceux qui manquent seront remplacés par des 00 dans le cas où tous ne sont pas indiqués explicitement.

c) Si on ignore la représentation en virgule flottante du nombre à empiler, l'image décimale de ce nombre devra être disposée dans des emplacements de mémoire. Expliquons-nous : supposons qu'on désire empiler 250. On placera à partir d'une adresse connue (nommons-la Ad) : $1E_h$ (code de « 2 »), 21_h (code de « 5 ») et $1C_h$ (code de « 0 »).

Ensuite, nous ferons appel à la routine spécialisée de la ROM qui lira ces caractères et effectuera la conversion.

Pour cela, il faut se souvenir que tous les caractères interprétés par le Basic sont lus grâce aux instructions RST 18 et RST 20 qui emploient la variable-système CH-ADD comme pointeur. Il s'avère donc nécessaire d'intervenir sur ce pointeur.

Enfin, on appellera au choix :
CALL 14D9 pour empiler un nombre réel ou
CALL 1548 pour empiler un nombre entier.

Finalement, nous aurons une routine de ce type (pour un nombre entier) :

LD HL, (CH-ADD) LD (Sauve), HL	Le pointeur CH-ADD est préservé en mémoire.
LD HL, Ad LD (CH-ADD), HL	CH-ADD reçoit maintenant l'adresse du premier caractère à interpréter.
RST 18 CALL 1548	Ce caractère est lu puis les suivants jusqu'à la rencontre d'un code ne représentant pas un chiffre.
LD HL, (Sauve) LD (CH-ADD), HL RET	La valeur étant empilée, CH-ADD est restauré avant le retour.

Une application de ce principe permet l'entrée de valeurs en décimal à partir du clavier dans le programme proposé en deuxième partie.

d) Enfin, il existe six zones (de 5 octets chacune) de la mémoire dont le contenu peut être chargé directement sur la pile par une instruction du calculateur (octets $E0_h$ à $E5_h$).

Cela correspond aux instructions RCL ou MR des petites calculatrices.

2) Les transferts à partir de la pile

Symétriquement avec ce qui existait pour les transferts précédents, on dispose de :

CALL 158A et **CALL 15CD** qui placent le contenu du sommet de la pile respectivement dans BC et A.

La paire BC ne peut recevoir que des entiers (positifs en l'absence de toute convention sur le signe du binaire) inférieurs à 65536, tandis que A vaut 255 au maximum. Il s'avère donc nécessaire d'arrondir les valeurs dépilées.

Si ces dernières excédent les limites rappelées ci-dessus, la retenue sera mise à 1.

Si ces valeurs sont positives, le drapeau Z sera levé (à 1).

Dans le sous-programme d'entrée de nombres (voir la deuxième partie), CALL 158A est utilisé pour récupérer dans BC la valeur hexadécimale de nombres frappés au clavier en décimal.

CALL 15DB inscrit sur l'écran, en décimal, de la même façon que le ferait un PRINT Basic, la représentation du nombre qui se trouve au sommet de la pile et qui est perdu à la suite de l'opération.

On emploiera cette routine pour afficher clairement le résultat d'un calcul.

Par exemple, dans le sous-programme présenté en seconde partie, une valeur disposée dans BC est imprimée sur l'écran en l'empilant d'abord (CALL 1520), puis en invoquant CALL 15DB.

Signalons enfin que les 6 zones de mémoire dont le contenu pouvait être chargé directement sur la pile serviront réciproquement à sauvegarder temporairement des valeurs. Des instructions spéciales du calculateur (octets $C0_h$ à $C5_h$) copient, en effet, le sommet de la pile dans la zone voulue. La pile n'est pas altérée.

Cela correspond aux STO ou M+ des petites calculatrices.

3) Les opérations unaires

Leur emploi reste des plus simples. Il suffit de ne pas oublier que les valeurs auxquelles elles s'appliquent sont perdues.

Exemple :

| A | → sin → | sin A | A est perdu.

pile

Voici les principales fonctions disponibles avec leur octet d'appel (après RST 28) ainsi que leur effet :

NEG	18_h	Négation (inversion du signe)
SIN	$1C_h$	Sinus
COS	$1D_h$	Cosinus
TAN	$1E_h$	Tangente
ASN	$1F_h$	Arc sinus
ACS	20_h	Arc cosinus
ATN	21_h	Arc tangente
LN	22_h	Logarithme népérien
EXP	23_h	Exponentielle
INT	24_h	Partie entière
RAC	25_h	Racine carrée
SGN	26_h	Signe
ABS	27_h	Valeur absolue.

Nous pensons que la manipulation des chaînes de caractères est plus aisée à partir du Basic. Voilà pourquoi nous n'indiquons pas les fonctions correspondantes.

4) Les opérations de composition

Tout aussi simples que les précédentes, ces opérations détruisent les 2 valeurs composées et réduisent la hauteur de la pile. Il faut se rappeler que l'opérateur A se trouve sur le sommet de la pile (dernier empilé) et l'opérande B juste au dessous :

SOUS	03_h	Soustraction : B — A
MUL	04_h	Multiplication : B × A
DIV	05_h	Division : B/A
PUIS	06_h	Puissance : B puissance A
ADD	OF_h	Addition : B + A.

5) Les utilitaires

a) Manipulation de la pile

ECH	01_h	Échange des 2 nombres placés au sommet de la pile : l'ordre B, A précédent devient A,B.
SUP	02_h	Supprime la dernière valeur du dessus de la pile : B,A donne B. A est perdu et la hauteur diminue de 5 octets.
DUP	$2D_h$	Duplique le nombre disposé au sommet de la pile : B,A aboutit à B,A,A, la hauteur s'accroissant de 5 cases.

b) Tests

Tous ces tests s'appliquent à la valeur située sur le dessus de la pile (A dans l'exemple choisi).

Cette valeur est détruite et l'on trouvera à sa place :
1 si la condition est remplie,
0 dans le cas contraire.

Nous ne citerons que 2 tests, les plus simples à employer et suffisants dans la plupart des cas :
INF-0 : 32_h donnera 1 si $A < 0$
SUP-0 : 33_h donnera 1 si $A > 0$

c) Les sauts

Comme en Basic, on peut sauter d'une instruction à une autre directement.

SAUT 2F $_h$ est un saut inconditionnel, exécuté dans tous les cas et qui n'affecte pas la pile.

SSV (saut si vrai) 00 $_h$ ne s'effectue que si le sommet de la pile est différent de zéro. On l'emploie d'ordinaire après un test. Dans ce cas, le saut sera pris si la condition est remplie. Le 1 ou le 0 laissé par le test au sommet de la pile est de toute façon détruit et la hauteur diminue.

Pour indiquer où conduit le branchement, un argument doit suivre l'instruction de saut (SAUT ou SSV). Le calcul de cet argument reste assez complexe :

— pour les sauts en avant :	SSV	
Argument :	03	Compter
	—	1
	—	2
Destination :	—	3
— pour les sauts en arrière :		Compter
Destination :	—	1
	—	2
	—	3
	SSV	4

255 — 4 = 251 soit FB $_h$

6) Exemple d'utilisation du calculateur pour un calcul complexe

La programmation du calculateur constitue une excellente initiation au travail sur pile et permettra d'aborder aisément des langages évolués du type de FORTH qui est promis à un bel avenir en micro-informatique.

Nous allons illustrer l'ensemble des propos qui précèdent par un exemple de calcul complexe employant plusieurs opérations successives.

Les non-mathématiciens pourront sauter cette partie.

a) Exposé du problème

NEWTON a démontré que la série définie par :

$$An = \frac{1}{2}\left(\frac{N}{An - 1} + An - 1\right)$$

converge vers la racine carrée de N. Ao peut être pris égal à N.

Nous souhaitons faire extraire au calculateur des racines carrées avec une précision de 10^{-10}.

b) Méthode

Deux zones de mémoire directement transférables à la pile seront employées :
M_1 conservera la valeur de N
Mo contiendra le dernier terme An. Puisque nous prenons Ao = N, le contenu de Mo sera N au début.

On calculera ensuite An + 1 par application de la formule précédente.

Puis on évaluera An + 1 — An. Si la différence est inférieure à 10^{-10}, nous imprimerons le résultat. Dans le cas contraire, un nouveau terme sera calculé.

c) Programme

Octets		Opérations	Pile	Commentaire
01-NN-NN		LD BC, N	....	N chargé dans BC
CD-20-15		CALL 1520	N	N empilé
EF		RST 28	N	Appel au calculateur
C0		STO M0	N	N en mémoire 0
C1		STO M1	N	N en mémoire 1
E0	Bouc	RCL M0	N, An	Calcul de An + 1
05		DIV	N/An	
E0		RCL M0	N/An, An	
0F		ADD	N/An + An	
A1		EMP-1	N/An + An, 1	
2D		DUP	N/An + An, 1, 1	
0F		ADD	N/An + An, 2	
05		DIV	(N/An + An)/2	= An + 1
2D		DUP	An + 1, An + 1	
E0		RCL M0	An + 1, An + 1, An	Calcul de la différence
03		SOUS	An + 1, An + 1-An	An + 1-An. Nous
27		ABS	An + 1, |DIF|	l'appelons DIF.
30		EMP-DONNEE		
CF-5B-E6		10^{-10}	An + 1, |DIF|, 10^{-10}	
FE-CC				
03		SOUS	An + 1, |DIF| - 10^{-10}	
32		INF-0	An + 1, 1 ou 0	DIF est-il inférieur à 10^{-10} ?
00		SSV	An + 1	Saut si vrai Là
06		Là	An + 1	Sinon nouveau calcul
C0		STO M0	An + 1	
02		SUP	----	
E1		RCL M1	N	
2F		SAUT		
E6		Bouc		
34	Là	FIN		Fin de calcul
C3-DB-15		JP 15DB	Affichage du résultat	

4. Les périphériques

Le ZX 81 communique avec le monde extérieur grâce à des périphériques qui sont : le clavier, l'écran de télévision, le magnétophone et l'imprimante.

A. Principes de base

Les transferts vers et à partir des périphériques s'opèrent grâce à des instructions du Z 80 spécialement conçues à cet effet. Il s'agit de OUT (pour les sorties) et IN (pour les entrées). Expliquons leur fonctionnement.

Nous savons que le microprocesseur peut envoyer par l'intermédiaire de 16 fils électriques (formant ce qu'on appelle un BUS) des adresses. Ces fils sont numérotés Ao à A15.

Dans le ZX 81, lors de la communication avec un périphérique, seules les lignes basses (Ao à A7) sont prises en compte (sauf dans le cas du clavier). Celle d'entre elles qui transmet un 0 binaire (c'est-à-dire qui se trouve portée à la masse électrique) distingue le dispositif concerné :

a) Ao = 0 sollicite le magnétophone en lecture (connexion EAR) ou bien le clavier. Ce dernier est connecté par huit fils. La partie haute de l'adresse doit donc être employée afin de scruter séquentiellement chacun d'entre eux.

Puisque seule Ao représente un 0 binaire, on peut figurer la moitié basse du BUS d'adresses par :

A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	1	1	1	1	0

soit en hexadécimal : FE_h.

Pour interroger le magnétophone ou le clavier, on aura donc recours à IN A, (FE) ou bien à IN A, (C), le registre C contenant FE_h.

L'accumulateur A recevra alors un octet (8 bits) représentatif de l'état de ces périphériques.

Deux bits revêtent une signification particulière :
— le plus à droite (bit 0) sera nul si et seulement si la touche BREAK est pressée ;
— le plus à gauche (bit 7) vaudra 1 si le signal provenant de la bande magnétique est haut (présence d'une impulsion).

b) A1 = O (donc moitié basse : 1111 1101 soit FD_h) communique avec le circuit maître, permettant de mettre en service ou de déconnecter le générateur d'interruptions. Nous ne nous servirons pas directement de cette possibilité.

c) A2 = 0 (donc moitié basse : 1111 1011 soit FB_h) accède à l'imprimante.

d) Enfin, si toutes les lignes d'adresses sont à 1 (1111 1111 ou FF_h), le magnétophone en enregistrement (connexion MIC) ou bien l'écran se verront concernés.

B. Le magnétophone

Voyons d'abord comment les programmes se trouvent inscrits sur la bande magnétique.

a) Représentation des données enregistrées

Le signal transmis au magnétophone peut être modulé grâce à des instructions d'entrées/sorties.

OUT (FF), A provoque une montée du signal tandis que IN A (FE) entraîne un état bas. Les 2 commandes limitent donc une impulsion électrique susceptible d'être inscrite sur bande magnétique. Le temps qui s'écoule entre OUT et IN détermine la largeur de l'impulsion.

Lors d'un « SAVE », il se passe d'abord une période de 5 secondes environ durant laquelle aucun accès au périphérique n'a lieu. La bande vierge défile donc et cela correspond au silence précédant un programme. Cette zone revêt plus

d'importance qu'on ne peut le penser. En effet, lors du chargement, la machine va « écouter » la bande et attendre qu'un silence suffisamment long (plus de 5/1000e de seconde) se manifeste. Elle saura alors que le début d'un enregistrement va suivre.

Après l'amorce, le titre du programme est transmis au magnétophone. Le dernier caractère de ce titre aura préalablement été mis en vidéo inversée.

Il convient ici de se souvenir qu'une lettre ou un chiffre en contraste inverse possède un code supérieur à 80_h. Par exemple, le code du « A » normal est 26_h et celui du « A » inversé, $A6_h$. Par conséquent le bit 7 (le plus à gauche) vaudra toujours 1 pour ces caractères (puisque $80_h = 10000000$ binaire). Cette particularité permettra, au chargement, de déceler la fin du titre.

Ensuite viendra le programme proprement dit, représenté par tous les octets d'adresses comprises entre les valeurs des variables-systèmes VERSN et E-LINE.

Seront ainsi copiés sur la bande :
— les variables-système elles-mêmes, à l'exception de quelques-unes (RAMTOP, ERR-SP),
— le texte Basic,
— le contenu de l'écran,
— les variables Basic.

Un octet est passé au magnétophone après découpage en ses huit éléments binaires, chacun de ceux-ci valant 1 ou 0.

Un 1 correspondra à 9 impulsions électriques tandis qu'un 0 n'en comportera que 4. Entre chaque série d'impulsions, un silence de durée égale à cinq d'entre elles sera respecté.

Exemple : l'octet A7 = 1010 0111 donnera sur la bande les trains d'impulsions suivants : 9-4-9-4-4-9-9-9-, les tirets représentant un délai équivalent à 5 impulsions.

b) La routine de chargement

Nous allons à présent nous pencher tout particulièrement sur la routine de chargement (LOAD) intégrée à la ROM. Les applications que nous en tirerons en deuxième partie de cet ouvrage s'avéreront extrêmement intéressantes.

Le lecteur qui ne désire pas approfondir cette étude peut passer les explications suivantes.

La routine LOAD relève d'une remarquable maîtrise de la programmation en langage machine et des possibilités particulières du Z 80.

Elle commence par un appel à la routine qui recherche le nom du programme à charger :

CALL 03A8

Ce sous-programme balaie la ligne Basic, vérifie qu'une chaîne de caractères est donnée (sinon erreur C), puis sélectionne le mode FAST, passe les paramètres de la chaîne depuis la pile du calculateur vers les registres DE (adresse de début de la chaîne) et BC (longueur), refuse une chaîne excédant 128 caractères (erreur F) et, enfin, met le dernier terme en contraste inverse. La retenue (carry) prend la valeur 0 si un nom a bien été fourni et 1 dans le cas contraire.

RL D
RRC D

copie le carry dans le bit 7 du registre D. Si un titre est donné, D vaut donc au moins 40_h puisque l'adresse de la chaîne représentant ce titre se situe en mémoire vive (débutant en 4000_h).

Si aucun nom n'est fourni, D vaut au moins 80_h puisque le carry (= 1 dans ce cas) a été transféré à son bit 7.

0347 CALL 34C
JR 347

La routine appelée ici (en $34C_h$) ramène dans le registre C les octets lus sur la bande magnétique. Cependant, si un silence de plus de 5 millisecondes survient, le retour vers le

programme appelant ne s'effectuera pas. On voit donc que la boucle ci-dessus va se poursuivre tant que la zone de silence précédant un enregistrement n'est pas atteinte.

034CLD C,01

C reçoit un marqueur dont la représentation binaire est 0000 0001. Par la suite, à chaque lecture d'un élément binaire, C se verra décalé à gauche et un 0 tombera dans le carry, le bit lu entrant dans ce registre par la droite. Ce n'est qu'au bout de 8 décalages que le carry recevra le 1 (marqueur) indiquant qu'un octet complet a été reconstitué. La routine sera alors achevée.

034E LD B, 00

Ceci sert à la temporisation de 5 millisecondes, B décomptant 256 boucles.

0350 LD A, 7F
IN A, (FE)

Le port d'entrée FE $_h$ est lu. Rappelons que son bit 7 vaudra 1 si un signal haut existe sur la bande et que son bit 0 passe à zéro si on presse BREAK.

OUT (FF), A

Renvoie à l'écran l'écho de ce qui est lu.

RRA
JR NC, 3A2

Le bit 0 est sorti par décalage à droite et s'il vaut 0, le BREAK sera pris en compte.

RLA
RLA
JR C, 385

Le bit 7 est à présent sorti par la gauche et, si un signal se manifeste, on va procéder à l'entrée.

DJNZ 350

Sinon, la boucle se poursuit dans les limites de 5 ms environ. Après quoi :

POP AF

L'adresse de retour au programme appelant est extraite de la pile.

CP D

0361 JP NC, Init

Tant que rien n'a été transféré depuis le magnétophone, D vaut au minimum 40 $_h$ et ce saut ne sera donc pas effectué. Par contre, si quelque chose a déjà été lu et qu'une interruption de signal de plus de 5 ms s'est produite, il s'avère nécessaire de réinitialiser le système (par un NEW) afin d'éviter une perte de contrôle.

LD H, D
LD L, E

L'adresse de début du titre demandé est passée dans HL.

0366 CALL 34C

Nouvel appel à la routine d'entrée des octets. Rappelons que le résultat de la lecture se retrouve dans le registre C.

BIT 7, D
LD A, C
JR NZ, 371

Si un nom de programme a été donné (le bit 7 de D vaut 0), les caractères qui le composent (pointés par HL) vont être comparés à ceux qui sont lus (passés de C dans A).

S'il n'y a pas eu de nom (le bit 7 de D vaut 1), n'importe quel programme convient et on saute en 371 $_h$.

CP (HL)
JR NZ, 347

Si le nom s'avère incorrect, on rentre de nouveau dans la boucle qui guette un silence.

0371 INC HL
RLA
JR NC, 366

Si l'un des caractères est correct, on passe au suivant, à moins qu'on n'ait atteint le dernier. Celui-ci est en contraste inverse, c'est-à-dire que le bit 7 de son code vaut 1. RLA sort ce bit et JR NC le teste.

INC (4015)

On va à présent charger le programme dans les octets compris entre 4009 $_h$ et l'adresse pointée par E-LINE.

E-LINE lui-même sera lu sur la bande magnétique. Mais, pour le moment, sa valeur est trop faible et le chargement risque de s'interrompre prématurément, d'où l'instruction précédente (4015 $_h$ correspond à la partie haute de E.LINE).

```
     LD HL, 4009
37B  LD D, B
```

D vaut désormais 0, indiquant qu'un chargement a eu lieu.

```
     CALL 34C
     LD (HL), C
     CALL 1FC
     JR 37B
```

Voici la boucle de chargement proprement dite. Le retour de la routine en 01FC $_h$ ne s'effectue que dans le cas où HL n'excède pas la valeur contenue dans E-LINE (même principe que pour le sous-programme débutant en 34C $_h$). La boucle va donc se poursuivre jusqu'à lecture complète.

```
385  PUSH DE
```

Ceci est la suite de l'écoute de la bande magnétique (voir plus haut) avec l'entrée des octets dans C. DE est d'abord préservé (surtout D en fait) pour servir comme nous l'avons vu.

```
     LD E, 94
388  LD B, 1A
38A  DEC E
```

On entre ici dans une double boucle qui décrémente E
— d'une part, tant que le signal reçu est haut,
— d'autre part (grâce au compteur B), tant qu'un intervalle silencieux (bas) de largeur suffisante, correspondant au silence enregistré entre deux trains d'impulsions, n'est pas détecté.

```
     IN A, (FE)
     RLA
```

Lecture du signal et transfert dans le carry.

```
     BIT 7, E
     LD A, E
```

Préparation à la sortie de la double boucle.

JR C, 388

1) Boucle si le signal est haut.

DJNZ 38A

2) Boucle si le signal est bas en décomptant la durée du silence.

Les temporisations sont calculées de manière que la double boucle est parcourue un peu moins de 17 fois pendant la durée d'une seule impulsion sur la bande magnétique (en moyenne 16,8 fois). Si l'on se souvient que 4 impulsions codent le bit 0 et 9 le bit 1, E est donc décrémenté (à peu près :

$4 \times 17 = 68$ fois pour un zéro
et $9 \times 17 = 153$ fois pour un 1.

Ce registre contenait $94_h = 148$ au début. Il reste donc environ 80 pour un zéro et un nombre négatif (c'est-à-dire dont le bit 7 vaut 1, en binaire signé) pour un 1.

POP DE
JR NZ, 39C

Le bit 7 de E a été testé avant la sortie des boucles. S'il s'avère non nul (cas d'un 1 lu) le saut s'effectue.

CP 56
JR NC, 34E

Pour un zéro lu, E contient environ 80. S'il excède 86 ($= 56_h$), la lecture est considérée comme erronée et rejetée.

Arrivé à ce point, le carry vaut :
0 pour un 1 lu (seule sortie possible de JR C, 388)
1 pour un 0 lu (seule sortie possible de JR NC, 34E).

39C CCF

L'inversion de la retenue rétablit la valeur correcte de la lecture.

RL C

Cette retenue est alors transférée dans le registre C par décalage.

JR NC, 34E

Et la lecture se poursuit tant que 8 décalages n'ont pas eu lieu (voir ci-dessus).

RET

Sinon, retour au programme appelant.

Un BREAK amène ici :

3A2 LD A, D
AND A
JR Z, 361

D est testé. Si D = 0, quelque chose a été chargé et il s'avère nécessaire de réinitialiser le système. Sinon un simple arrêt sur le compte rendu D s'opère :

RST 08
0C.

Le schéma général de ce talentueux sous-programme sera intégralement repris dans deux routines données en seconde partie. La première lit les titres des programmes enregistrés et les affiche à l'écran, la seconde vérifie la qualité des sauvegardes sur cassette.

C. L'imprimante

Le manuel d'utilisation qui accompagne l'imprimante ZX Printer n'est pas très explicite sur la manière dont on procède pour travailler en haute résolution. Voici quelques informations complémentaires.

Plutôt que de chercher à composer un programme gérant complètement l'imprimante, il s'avère préférable de reproduire dans un bloc de mémoire vive la routine « COPY » de la ROM.

Cette dernière peut en effet être implantée à n'importe quelle adresse sans nécessiter de modifications.

Elle se trouve en 869_h-$8E1_h$ (2153-2273) et on peut y entrer à partir de trois adresses :

— 869_h provoquant la recopie des 22 premières lignes de l'écran ;

— 871_h envoyant à l'imprimante le contenu du tampon (adresses 16444-16476 ou $403C_h$-$405C_h$) ;

— 876_h copiant D fois 32 caractères + 1 N/L (par exemple si

D contient 5, 5 lignes de 32 caractères + N/L, qu'il ne faut surtout pas oublier, seront passées à l'imprimante).

La solution préconisée par Sinclair consiste à se servir de de la seconde entrée.

On reproduira donc en mémoire vive (par exemple à partir de 16514) le contenu des adresses 2161 (871 $_h$) à 2273, soit 112 octets :

```
 1 REM suivi de 113 espaces,
 5 LET Q = 16514
10 FOR X = 2161 T0 2273
15 POKE Q, PEEK X
20 LET Q = Q + 1
25 NEXT X.
```

Il manque encore un RET final car il n'existe pas dans la ROM. On ajoutera donc :

30 POKE Q, 201.

Les motifs transmis à l'imprimante sont pris dans la table bien connue située à partir de l'adresse 7680 (1E00 $_h$).

L'adresse des motifs représentant un caractère de code c se calcule par : 7680 + 8 x c.

Voici par exemple la liste des 8 octets formant le motif du A :

```
HEXA    BINAIRE
00      ........
3C      ..1111..
42      .1....1.
42      .1....1.
7E      .111111.
42      .1....1.
42      .1....1.
00      ........

NOTE  :  LES 0 DU BINAIRE SONT
REPRESENTES PAR DES "." POUR
FACILITER LA LECTURE.
```

On peut obtenir que le système aille prendre ses motifs à une autre adresse. Il existe en effet dans la routine que nous avons recopiée une instruction :

LD H, OF

L'octet OF_h se trouve 56 cases après le début du sous-programme. Ainsi, si on a reproduit la routine en 16514, OF_h se situe en 16514 + 56 = 16570.

Or ce OF_h représente la moitié de $1E_h$ (partie haute de l'adresse $1E00_h$ = 7680).

Supposons que l'on souhaite redéfinir les motifs binaires à partir de l'adresse 24576 (= 6000_h), il suffira de faire :

```
35 POKE 16570, 48
```

car 48 = 30_h (la moitié de la partie haute de 6000_h).

A présent, nous pouvons réaliser le dessin de nos nouveaux caractères, celui de code 00 étant représenté à partir de 24576, celui de code 01 à partir de 24584, etc. On a ainsi accès au point élémentaire.

Pour faire apparaître les motifs ainsi définis sur l'imprimante, on disposera par des POKE les codes correspondants dans le tampon et on appellera la routine par RAND USR 16514.

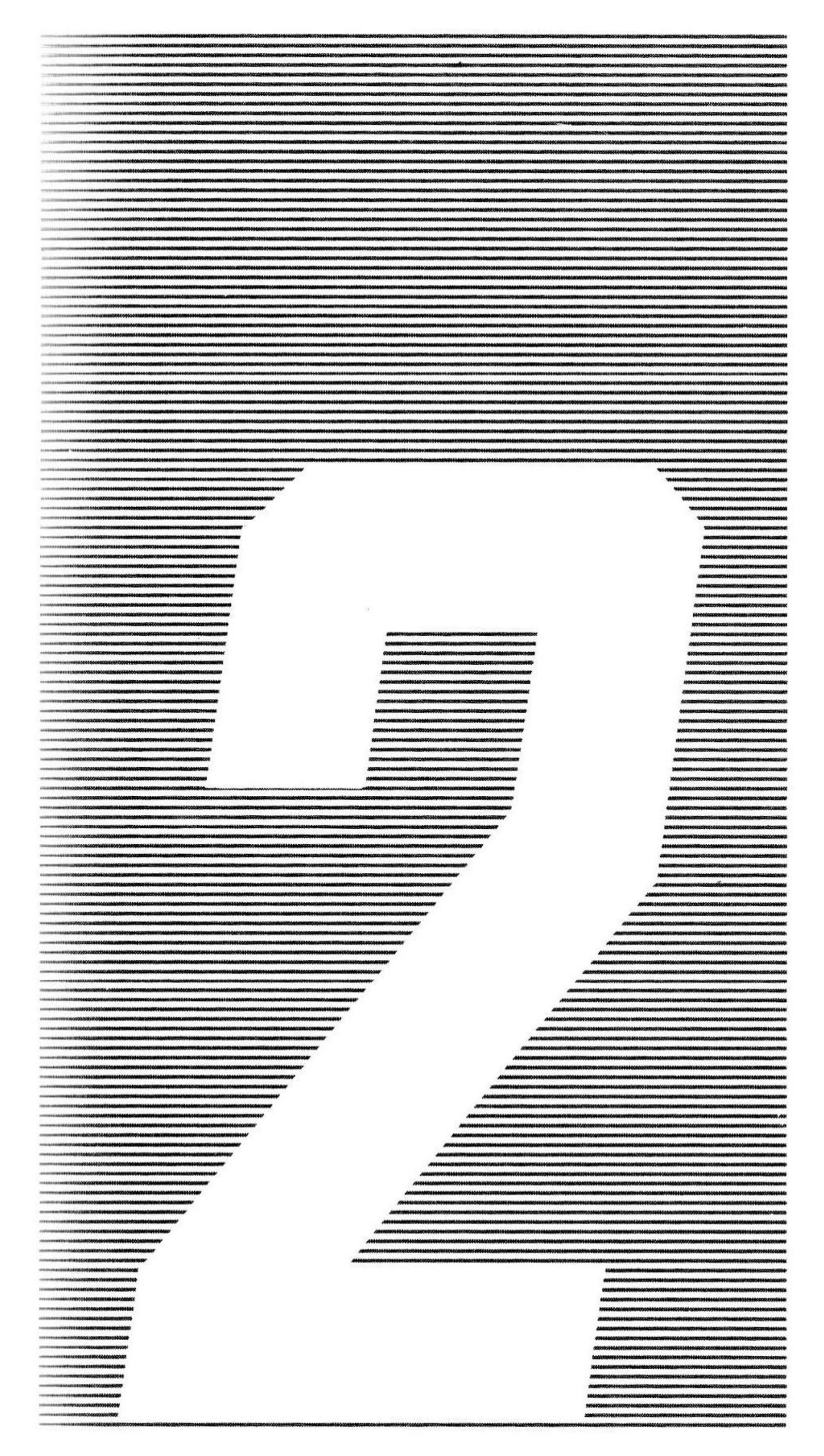
Dans l'exemple suivant, nous avons dessiné un alphabet minuscule :

abcdefghijklmnopqrstuvwx yz ¯àéè '█

Pour cela, après avoir redessiné tous les caractères à partir de 24576, comme indiqué plus haut, nous avons employé :

```
40 LET Q = 16444         (début du tampon)
45 FOR X = 0 T0 31
50 POKE Q, X
55 LET Q = Q + 1
60 NEXT X
65 RAND USR 16514
```

Il est bien sûr possible de redéfinir les dessins par calcul et d'obtenir ainsi des sorties en haute résolution. Mais les calculs s'avèreront toujours longs et complexes.



Applications

Nous allons à présent considérer quelques applications des principes que nous avons exposés jusque-là. Il s'agit d'une série d'utilitaires disposés dans les adresses hautes de la mémoire. Il conviendra de protéger cette zone par modification de RAMTOP selon le procédé classique.

Chacun des utilitaires peut être employé séparément, seuls quelques petits modules (scrutation du clavier, entrée des nombres, etc.) étant communs. Mais l'ensemble des fonctions constitue un logiciel performant et extrêmement utile.

Lorsqu'on aura introduit en machine l'une des routines, on ne l'expérimentera qu'après avoir contrôlé qu'elle ne comporte pas d'erreurs. Pour cela, on emploiera :
— la somme de contrôle donnée (somme des octets),
— éventuellement le désassembleur,
— la redondance (c'est-à-dire la répétition) résultant de la présence simultanée, sur la liste fournie, des codes hexa et des mnémoniques. En comparant les uns et les autres et en se reportant à l'annexe page 181 du manuel du ZX 81, on pourra s'assurer de déchiffrer parfaitement ce qui est imprimé.

Sauf indication contraire, tous ces programmes sont conçus pour fonctionner en mode SLOW.

En premier lieu, nous avons jugé utile de donner la liste du désassembleur (en Basic) qui a servi à imprimer tous les listings en langage assembleur.

1. Le désassembleur

Ce programme utilise des tableaux qu'il faut initialiser avant de frapper la liste proprement dite. On emploie pour cela un petit programme :

```
 5 DIM A$ (256,10)
10 FOR K = 1 T0 256
15 INPUT A$ (K)
20 PRINT K ; « » ; A$ (K)
25 NEXT K
```

Il y a 5 tableaux dont les dimensions sont indiquées ci-après :

A$ (256,10)
B$ (10,4)
C$ (58,13)
F (14)
G (25).

Tel qu'il apparaît ci-dessus, le chargeur ne permet que l'initialisation de A$. Une fois qu'on aura introduit ce premier tableau, on modifiera le programme de manière à remplir les autres tables. Ainsi B$ sera initialisé après les modifications suivantes :

```
 5 DIM B$ (10,4)
10 FOR K = 1 T0 10
15 INPUT B$ (K)
20 PRINT K ; " " ; B$ (K)
25 NEXT K
```

Pour modifier la liste, utiliser EDIT et surtout pas NEW car toutes les valeurs déjà introduites seraient perdues.

Attention : Dès que l'un des tableaux ou même quelques-uns de ses éléments seulement sont en mémoire, on ne DOIT PLUS UTILISER LA COMMANDE RUN, sinon tout serait effacé. On emploiera, selon le cas, CONT pour poursuivre l'exécution d'un programme ou GOTO 1 pour commencer cette exécution.

Au fur et à mesure que l'on rentre les valeurs des éléments des tableaux, ces valeurs sont présentées sur l'écran en même temps que le numéro d'ordre de l'élément. Si l'on constate qu'on a commis une erreur, on la corrigera lorsque le programme s'arrêtera sur le compte rendu 5 (écran plein). Ensuite, on reprendra par CONT. Par exemple : si, pour A$ (4) on a rentré INC B au lieu de INC BC, lorsque l'écran est plein, on fera en commande directe :
LET A$ (4) = « INC BC »

Puis on frappe CONT. S'il n'y a aucune erreur, on fait directement CONT.

Contenu des tableaux

A$ contient les mnémoniques de la colonne « Assembleur Z80 » qui se trouvent à partir de la page 181 dans le manuel du ZX 81. Toutefois, on remplacera N par $ et DIS par £.
Ainsi aura-t-on : A$ (1) = « NOP » ; A$ (2) = « LD BC, $$ » ; etc.
A$ (25) = « JR £ » ; etc.
Pour A$ (204) ; A$ (222) ; A$ (238) ; A$ (254) qui tombent sur des préfixes (CB, DD, ED, FD), on mettra comme valeur un blanc (espace).

B$ contient les instructions types de la colonne « Après CBh », c'est-à-dire : RLC, RRC, RL, RR, SLA, SRA, SRL, BIT, RES et SET.

C$ contient les instructions de la colonne « Après ED h » présentées comme pour A$ mais précédées de la valeur en décimal sur *3 chiffres* de leur code. On trouvera donc successivement : 064IN B, (C) 065OUT (C), B 066 SBCHL, BC 067LD ($$), BC 068NEG, etc.

F contient : 9 25 33 34 35 41 42 43 57 225 227
229 233 249

G contient : 52 53 54 70 78 86 94 102 110 112
113 114 115 116 117 119 126 134 142
150 158 166 174 182 190

La présentation des mnémoniques sur l'écran dépend de la manière dont les tableaux ont été initialisés. On pourra disposer des blancs aux endroits propices sans toutefois excéder les dimensions autorisées pour chaque table. Ainsi A$ (43), par exemple, pourra être initialisé à « LD_HL, ($$) » mais non pas « LD_HL, _ ($$) » qui fait 11 caractères alors que A$ est dimensionné à (256,10). De même, dans B$, on pourra laisser un blanc après chaque mnémonique type. Par exemple B$ (1) = « RLC_ ».

Lorsque la fastidieuse tâche précédente est achevée, il est

recommandé de sauvegarder les variables sur cassette pour parer à toute perte de contrôle.

Le désassembleur doit ensuite être introduit (sans faire NEW, les nouvelles lignes remplaçant les anciennes au fur et à mesure).

L'utilisation du désassembleur demeure très simple :
Frapper GOTO 1 (N/L)

Le programme demande alors où doit commencer le désassemblage. Trois réponses sont possibles :
— « S » (pour « suite »), qui affiche la suite de ce qui se trouve sur l'écran ;
— un nombre en décimal (exemple : 2150) ;
— une valeur en hexadécimal terminée par un « H » (exemple : 3 E4H).

Après N/L, l'écran est effacé et 21 lignes désassemblées apparaissent avant de retourner à la question « ADRESSE ? ».

Notons d'une part que toutes les données affichées sont en hexadécimal et d'autre part que la ligne : 1 REM reste disponible afin que l'on puisse y développer un programme en langage machine tout en conservant le désassembleur.

2. Fonctions multiples

(voir organigramme pages 120-121).

Étudions à présent les routines en assembleur.

Celles-ci réalisent un certain nombre de tâches utiles. Nous verrons un peu plus loin qu'un « aiguillage » permet de choisir la fonction désirée en pressant une simple touche.

Le premier petit module, utilisé en conjonction avec la routine d'aiguillage, permet d'appeler successivement plusieurs fonctions. Normalement, chacune d'elles s'achève par un RET qui renvoie au Basic. Par contre, si on frappe « X » suivi de N/L, en tout premier lieu, une série de commandes pourra être employée.

Pour cela, une adresse disposée sur la pile de la machine renverra en 78FC $_h$. Un appel à la routine de scrutation du clavier (7E81 $_h$) placera alors la machine en attente jusqu'à ce qu'une touche (par exemple N/L) soit pressée. A ce moment, CALL 0A2A effacera l'écran et un saut à la routine d'aiguillage s'effectuera.

Modules communs utilisés

7E81 $_h$: Scrutation du clavier.
7918 $_h$: Aiguillage.

Somme de contrôle

De 78FC $_h$ à 7907 $_h$ (30972-30983) : 1391

3. Retour au Basic
(commande « B »)

Dans le cas où l'on utilise le module précédent, le retour au Basic ne pourra s'obtenir qu'en vidant la pile des retours de sous-programmes. Cela est possible grâce à RST 08 et la machine affiche un code d'erreur E (OD $_h$ + 1).

Somme de contrôle 220

4. Tables

La table 790A $_h$ — 7917 $_h$ contient les codes des initiales des différentes commandes (T, S, M, G, E, H, O, D, K, C, V, R, B, X).

La table 7944 $_h$ — 795F $_h$ conserve les adresses de début de chacun des modules, dans le même ordre que les lettres dans la liste précédente.

La troisième colonne de la liste désassemblée n'a ici aucune signification.

5. Aiguillage

Ce module commence en 7918_h (31000).

A partir de l'initiale d'une commande (trouvée dans la première table), il aiguille vers l'adresse correspondante (prise dans la seconde).

Il suffit donc de faire :
RAND USR 31000
pour accéder à toutes les fonctions. C'est pourquoi on a choisi une adresse simple à retenir.

Chaque fonction sera appelée par une lettre de commande. Il ne sera pas nécessaire de presser N/L pour terminer une commande sauf pour X (fonctions multiples).

Fonctionnement et description

Au lancement de la routine, le système est placé en mode SLOW (CALL 0F2B) et le signe > indique que l'initiale d'une commande est attendue.

La scrutation du clavier (CALL 7E81) ramène le code de la touche pressée. Ce code est recherché dans la première table et refusé s'il ne s'y trouve pas.

Par contre, lorsqu'une commande valide est fournie, son initiale s'affiche et l'adresse du module en langage machine correspondant est trouvée dans la seconde table.

Le saut final (JP (HL)) provoque l'exécution de la fonction.

Voici les lettres acceptées et les fonctions obtenues :

T : Transfert en REM
S : Suppression des REM
M : Moniteur hexadécimal
G : Générateur de REM longues
E : Élimination de lignes Basic
H : Conversion en hexadécimal
O : Nombre d'Octets libres

D : Conversion en Décimal
K : Titres des programmes enregistrés sur cassette (K7)
C : Contrôle des enregistrements
V : Liste des Variables
R : Renumérotation
B : Retour au Basic
X : Fonctions multiples.

Ces deux derniers modules (B et X) ont déjà été décrits, les autres le sont plus loin.

Modules communs utilisés

7E81 $_h$: Scrutation du clavier.

Somme de contrôle

De 7918 $_h$ à 7943 $_h$ (31000-31043) : 4186.

6. Transfert en REM

Le moniteur d'hexadécimal donné par ailleurs facilite beaucoup la mise au point de programmes en langage machine directement à partir de l'adresse 16514 (dans une ligne : 1 REM). Cependant, on peut souhaiter développer ce genre de programme en d'autres emplacements de la mémoire. Le problème de la sauvegarde sur cassette se pose alors.

Le module proposé permet le transfert d'un bloc de mémoire dans une REM à la ligne 1.

La commande « SAVE » du Basic est alors applicable.

A la lecture, il s'avère aisé de retransférer le bloc à sa place d'origine grâce à une boucle en Basic.

Fonctionnement

Avant d'appeler la routine de transfert, entrer une ligne : 1 REM.

Faire ensuite RAND USR 31000 et presser « T » en réponse au prompt (>).

Un point d'interrogation indique que la machine attend que l'on précise quel bloc de mémoire on désire transférer.

Par exemple, si on veut disposer dans la REM les contenus des adresses 300O0 à 30500, on répondra :
?30000, 30500.

Noter la virgule séparant les deux valeurs et le point final. Ce format s'appliquera chaque fois que l'un des modules exigera 2 paramètres.

A la suite du point final, l'écran clignote : le transfert s'est opéré.

Description

Un appel à la routine d'entrée de 2 nombres en décimal ($7E31_h$) a d'abord lieu. Ces 2 nombres sont chargés dans HL et DE. La longueur de la REM nécessaire est calculée et passée dans BC. La routine de création de REM longues (en $7BC5_h$) est ensuite invoquée afin d'obtenir la place indispensable. Enfin, le transfert ne demande plus qu'une instruction automatique LDIR.

Modules communs utilisés

$7E31_h$: Entrée de 2 nombres en décimal.
$7BC5_h$: Création de REM longues.

Somme de contrôle

De 7960_h à $797C_h$ (31072-31100) : 3848.

7. Suppression des REM

Ce module permet de compacter les programmes Basic en supprimant les REM.

Les sauts (GOTO et GOSUB) n'ont pas besoin d'être modifiés même s'ils renvoyaient à une ligne REM. En effet, le Basic du ZX 81 offre la particularité suivante : si une ligne n'existe plus, le branchement se fera vers la suivante.

Fonctionnement

Faire RAND USR 31000 suivi de « S ». Le nombre d'octets restant libres avant et après élimination des REM s'affiche, ce qui permet de juger de la place gagnée.

Description

La routine de calcul du nombre d'octets libres ($7C18_h$) est appelée deux fois : au début et lorsque le travail est terminé. Après le premier appel, un > est imprimé afin de séparer les deux valeurs. Ensuite le programme Basic est balayé à partir de son début ($407D_h = 16509$). Sa fin est détectée grâce à l'octet 76_h (N/L).

La routine commençant en $7FE0_h$ passe dans A le code de la première instruction Basic de chaque ligne. S'il s'agit de REM (EA_h), un appel au sous-programme d'élimination des lignes ($7BF5_h$) supprime l'indésirable.

Modules communs utilisés

$7C18_h$: nombre d'octets libres,
$7FE0_h$: passe dans DE la longueur de la ligne Basic et dans A le code de la première instruction,
$7BF5_h$: élimine des lignes.

Somme de contrôle

De 797D $_h$ à 799E $_h$ (31101-31134) : 4262.

8. Moniteur d'hexadécimal

Voici à présent un moniteur extrêmement pratique et rapide à charger. Il est recommandé de l'introduire en tout premier lieu à partir de l'adresse 31135 et d'y ajouter les modules communs qu'il utilise.

On pourra alors l'employer (sans passer par la routine d'aiguillage) en frappant : RAND USR 31158. Il servira à implémenter les autres modules proposés. On n'oubliera pas, bien sûr, de protéger les emplacements de mémoire nécessaires par modification de RAMTOP.

Fonctionnement

Frapper RAND USR 31000 puis « M » (en présence du module d'aiguillage). L'entrée dans le moniteur se manifeste par l'affichage d'une étoile. On doit alors donner une adresse indiquant la zone de mémoire où l'on désire travailler (adresse en hexadécimal). Les 4 derniers chiffres hexa sont pris en compte. Presser N/L pour afficher la « page » de travail. Celle-ci comprend :
— en haut à droite, l'adresse de l'octet en cours (pointé par le curseur),
— 20 lignes de 8 octets (160 octets) avec, au début de chacune, l'adresse correspondante.

Le curseur (demi-octet en contraste inverse) apparaît sur le premier octet.

On peut travailler dans toutes les zones de mémoire vive mais la zone commençant en 4082 $_h$ (16514) est tout à fait recommandée. En effet, si l'on crée une ligne Basic présentée comme suit : 1 REM, l'adresse 16514 correspond

à l'octet qui suit le mot clef REM. Nous verrons que la commande G décrite plus loin permet de réserver la place désirée après ce mot clef.

D'autre part, un programme en langage machine constitué dans cette ligne pourra être préservé par la commande Basic « SAVE ». Enfin, seule cette zone de mémoire permettra d'utiliser les fonctions d'insertion/suppression du moniteur sans risque de perte de contrôle.

Commandes disponibles sous moniteur

Lorsque la page de travail est affichée et à condition que le curseur soit apparent, on dispose de :
— Q pour quitter le moniteur et retourner au Basic (sauf si la commande X est en cours, auquel cas l'écran est effacé et on peut appeler une nouvelle fonction),
— BREAK (espace) pour effacer la page et pouvoir en demander une autre.

• *Curseur :* Les 4 flèches (SHIFT avec 5, 6, 7 ou 8) permettent, si le curseur est visible, de le déplacer dans le sens voulu pour atteindre n'importe quel octet. Les déplacements en dehors des limites de la page sont possibles mais provoquent une réactualisation de l'écran. La nouvelle page reprend à partir de l'adresse où vient théoriquement le curseur en quittant l'écran.
Le compteur en haut et à droite indique à tout moment l'adresse de l'octet en cours.

• *Modification d'octets :* On peut modifier l'octet pointé par le curseur. Il suffit de frapper les chiffres hexadécimaux voulus. Après le premier chiffre, le curseur disparaît. Il se montre à la case suivante après le second. Pendant l'extinction du curseur, aucune commande n'est possible. Tout caractère qui ne représente rien en hexadécimal est refusé.

• *SHIFT GRAPHICS :* Lorsqu'on presse ces touches, le curseur s'éteint. Le code Sinclair de la première touche

frappée ensuite s'inscrit dans l'octet en cours et le curseur passe au suivant.

Exemple :

```
                                   00 00
   GRAPHICS                        00 00
          A                        25 00
```

• *SHIFT RUBOUT :* Ces touches permettent de calculer les déplacements (offset) pour les sauts relatifs. Déplacer d'abord le curseur pour le positionner sur l'octet où doit aboutir le saut. Frapper SHIFT RUBOUT. Revenir ensuite sur la case où doit s'inscrire le déplacement. En frappant de nouveau SHIFT RUBOUT, l'offset apparaît dans cette case. Si le saut relatif s'avère impossible (hors de portée), RUBOUT n'a aucun autre effet que d'effacer le curseur.

Exemple : on désire sauter de la dernière case de la ligne à celle qu'indique l'étoile :

```
           00  00  00  00  00  00  18  00
           *
<---       00  00  00  00  00  00  18  00
RUBOUT     00  00  00  00  00  00  18  00
--->       00  00  00  00  00  00  18  00
RUBOUT     00  00  00  00  00  00  18  FA
```

• *SHIFT FUNCTION :* Quand le curseur est affiché, SHIFT FUNCTION permet d'ouvrir un emplacement (pour insertion) en repoussant tous les octets à partir du curseur vers le haut.

• *SHIFT EDIT :* Si le curseur est visible, cette manœuvre supprime l'octet en cours en redescendant tous ceux qui se trouvent à des adresses supérieures.

Pour ces deux dernières fonctions, il convient de noter que la page reprend à partir de l'adresse à laquelle se situait le curseur. Ceci autorise les insertions/suppressions en série. D'autre part, ces possibilités ne doivent être employées que si l'on travaille dans la page commençant en 4082_h, dans la ligne 1 REM. On ne doit jamais effacer le N/L final de la ligne

(code 76$_h$). Attention aux déplacements des instructions de sauts qui peuvent n'être plus valables après insertion ou suppression.

• *NEWLINE :* Si le curseur est apparent, N/L affiche au centre de la ligne supérieure de l'écran le code binaire de l'octet en cours.

• *SHIFT G :* (Go) Exécute le code machine à partir de l'octet où se trouve le curseur. Au retour (par RET), les registres principaux du Z 80 sont affichés.

• *SHIFT R :* Présente sur les 2 lignes inférieures de l'écran le contenu de certains registres du microprocesseur au moment où l'on presse les touches.

A la fin de ces deux commandes, seul BREAK est accepté et renvoie à l'étoile du moniteur.

Description

Voici un bref commentaire sur la structure du programme :

79B6$_h$ - 79CO$_h$: Effacement de l'écran et appel de la routine affichant l'étoile et demandant l'adresse de début (terminée par N/L).

79C1$_h$ - 79C8$_h$: Passage de l'adresse dans HL et sauvegarde dans deux cases de mémoire.

79C9$_h$ - 79E1$_h$: Affichage de la page de travail.

79E2$_h$ - 79EF$_h$: Inscription du curseur par inversion vidéo du premier demi-octet.

79F0$_h$ - 79FC$_h$: Affichage de l'adresse de l'octet en cours en haut et à droite de l'écran. La prochaine position d'écriture est ensuite ramenée au niveau de l'octet en cours.

79FD$_h$ - 7A0E$_h$: Boucle de surveillance du clavier. Détection de BREAK renvoyant au début et de « Q » relançant le Basic.

7A0F$_h$ - 7A2A$_h$: Modification d'un octet et passage au suivant.

7A2B$_h$ - 7A43$_h$: Test des limites à droite et en bas de l'écran. Si la limite droite est atteinte, passage à la ligne suivante. Si le bas de l'écran est dépassé, production d'une nouvelle page.

7A44$_h$ - 7A4F$_h$: Préparation aux fonctions spéciales. Extinction du curseur.

7A50$_h$ - 7A65$_h$: Routine curseur à droite.

7A66$_h$ - 7A84$_h$: Routine curseur à gauche. Test des limites à gauche et en haut de l'écran.

7A85$_h$ - 7A99$_h$: Routine curseur en bas.

7A9A$_h$ - 7AAE$_h$: Routine curseur en haut.

7AAF$_h$ - 7ABE$_h$: Entrée de codes de caractères (fonction SHIFT GRAPHICS).

7ABF$_h$ - 7AFB$_h$: Calcul des déplacements pour les sauts relatifs (fonction SHIFT RUBOUT). Le bit 7 de l'octet d'adresse 4021$_h$ sert de drapeau pour distinguer l'enregistrement d'une adresse (première pression de RUBOUT) et le calcul de l'offset (seconde pression).

7AFC$_h$ - 7B0D$_h$: Fonction d'insertion (SHIFT FUNCTION).

7BOE$_h$ - 7B27$_h$: Fonction de suppression (SHIFT EDIT).

De 7B15$_h$ à 7B27$_h$, la longueur de la ligne 1 REM est corrigée après insertion/suppression. C'est à cause de cette correction qu'on ne peut rien insérer ou supprimer dans une page de travail située en dehors de la REM.

7B28$_h$ - 7B4E$_h$: Routine inscrivant le code binaire de l'octet en cours au milieu de la ligne du haut de l'écran. On notera comment la valeur de S-POSN (position d'écriture courante) est sauvegardée sur la pile puis retrouvée par CALL 0918.

7B4F$_h$ - 7B63$_h$: Première partie de la routine d'écriture du contenu des registres : sauvetage des registres sur la pile.

7B64$_h$ - 7B6D$_h$: Table des noms des registres (la colonne des mnémoniques n'a aucune signification).

7B6E$_h$ - 7B8E$_h$: Fin de la routine d'affichage des contenus des registres.

7B8F $_h$ - 7BA2 $_h$: Sous-programme lançant l'exécution de routines en langage machine (SHIFT G). Noter en 7B9D l'instruction CALL qui réalise le lancement. L'argument de ce CALL est modifié par le programme. La somme de contrôle indiquée plus loin correspond à CALL 4082.

7BA4 $_h$ - 7BA9 $_h$: Sous-programme modifiant l'adresse de l'octet en cours lors des déplacements du curseur. La nouvelle adresse vaut (HL) + (DE), DE contenant éventuellement un nombre négatif (binaire signé).

7BAA $_h$ - 7BAD $_h$: Sous-programme de copie de la valeur de A dans l'octet en cours.

7BAE $_h$ - 7BB5 $_h$: Passage à l'octet suivant.

7BB6 $_h$ - 7BC1 $_h$: Routine d'inscription, en haut et à droite de la page, de l'adresse de l'octet en cours.

Modules communs utilisés

7C41 $_h$: Entrée d'une adresse hexadécimale dans BC.
7C61 $_h$: Utilitaire de décalage également appelé par le sous-programme précédent.
7D3D $_h$: Affichage du contenu de HL.
7D49 $_h$: Affichage du contenu de A (également appelé par la routine précédente).
7E81 $_h$: Scrutation du clavier.

Somme de contrôle

De 799F $_h$ à 7BC1 $_h$ (31135-31681) : 62703.
(Attention : cette somme n'est valable que s'il y a CALL 4082 en 7B9D $_h$).

9. Générateur de REM longues

Cette fonction crée une REM à la ligne 1, ayant la longueur désirée, afin d'y développer des routines en langage machine.

Fonctionnement

Entrer d'abord, sous Basic : 1 REM (N/L).

Ensuite, faire RAND USR 31000, puis « G ».

Répondre au point d'interrogation en indiquant le nombre d'octets à réserver (en décimal).

Exemple : ?300.

Noter le format de la réponse. Chaque fois qu'une fonction n'exigera qu'un seul paramètre, ce dernier sera terminé par un point. Si, par erreur, on a frappé une virgule, il suffit d'introduire un point final immédiatement après pour obtenir l'exécution normale de la commande.

A la dernière pression de touche, l'écran clignote et la REM est créée, les octets libérés contenant 16_h (code du tiret).

Attention : Un défaut de conception *du Basic* et non pas du programme proposé fait que, lorsqu'une REM contient plus de caractères que l'écran n'en peut représenter, il est nécessaire de disposer une ligne 2 REM qui sert de tampon. Sinon, lors d'une liste automatique, on risque de perdre le contrôle du système.

Description

L'appel à la routine d'adresse $7C11_h$ permet de saisir au clavier un seul nombre en décimal. La valeur hexadécimale de ce nombre est récupérée dans BC. HL pointe sur l'emplacement à partir duquel la place doit être dégagée. (4082_h = 16514, soit juste après le mot clef REM).

Un passage provisoire en mode FAST (CALL 02E7) précède l'appel de la routine intégrée à la ROM chargée de déplacer les contenus des blocs de mémoire.

Ensuite, la longueur de la ligne 1 est corrigée.

Enfin, une boucle utilisant BC comme compteur charge un code 16_h (code de « - ») dans tous les emplacements libérés.

Le saut final en 207_h rétablit le mode SLOW.

Modules communs utilisés

$7C11_h$: Entrée d'un seul nombre en décimal.

(Attention : ce module n'est qu'une variante de celui qui débute en $7E31_h$ et ne peut donc pas fonctionner tout seul).

Somme de contrôle

De $7BC2_h$ à $7BE3_h$ (31682-31715) : 3512.

10. Élimination de lignes Basic

Ce module permet de supprimer simultanément plusieurs lignes consécutives d'un programme Basic.

Fonctionnement

L'accès à cette fonction s'obtient par :
RAND USR 31000 suivi de « E ».

Lors de l'affichage du point d'interrogation, il convient de donner les numéros de la première et de la dernière ligne à éliminer. Le format reste celui que nous avions expliqué au paragraphe 6 (« Transfert en REM »).

Exemple : ?1, 10.
éliminera les lignes dont les numéros vont de 1 *inclus* à 10 *exclu*. Si la ligne 10 est la dernière du programme et qu'on désire la supprimer aussi, on peut frapper :

? 1,11.

Lorsqu'on entre le point final, l'écran clignote un instant.

Description

Nous avons vu (paragraphe 6) que la routine commençant en 7E31 $_h$ permet de prendre au clavier deux nombres décimaux séparés par une virgule et achevés par un point.

Le premier numéro de ligne est chargé dans HL et l'appel au sous-programme en 9D8 $_h$ va ramener dans ce même registre l'adresse de début de la ligne en question.

Après sauvegarde de HL sur la pile, le même traitement est appliqué au second numéro.

DE reçoit ensuite la valeur de HL précédemment empilée.

Il ne reste plus alors qu'à passer en mode FAST avant d'appeler la routine spécialisée de la ROM et de retrouver enfin le mode SLOW.

Modules communs utilisés

7E31 $_h$: Entrée de 2 nombres en décimal.

Somme de contrôle

De 7BE4 $_h$ à 7BFD $_h$ (31716 - 31741) : 2966.

11. Conversion en hexadécimal

Cette fonction donne la conversion hexadécimale des nombres décimaux valant 1 à 65535.

Fonctionnement

Frapper RAND USR 31000 puis « H ».

La machine réclame la valeur à convertir. Le format est celui décrit au paragraphe 9 (« Générateur de REM »).

L'entrée du point final provoque l'affichage du signe "=" et de la valeur hexadécimale correspondante.

Exemple : ?65535. = FFFF

Pour quitter cette fonction, presser simplement un point sans introduire préalablement un nombre.

Noter que si la commande X est en cours, il faudra encore appuyer sur une touche (par exemple N/L) pour employer un autre module.

Description

Il s'agit d'une simple utilisation des ressources déjà disponibles. La routine d'adresse $7C11_h$ saisit la valeur décimale qui se trouve immédiatement convertie en hexa dans BC. Ce registre est copié dans HL et un appel au sous-programme affichant HL est effectué après l'inscription du signe "=".

Enfin, on passe à la ligne d'écran suivante (par LD A, 76 et RST 10) et, si la dernière valeur inscrite est nulle un retour a lieu. Sinon on procède à une nouvelle entrée.

Modules communs utilisés

$7C11_h$: Entrée d'un seul nombre en décimal.
$7D3D_h$: Affichage à l'écran du contenu de HL.

Somme de contrôle

De $7BFE_h$ à $7C10_h$: (31742-31760) : 2388.

12. Entrée d'une seule donnée décimale

Nous expliquerons plus loin comment la routine commençant en 7E31 $_h$ autorise la saisie de 2 nombres décimaux au clavier. Le même programme est employé pour l'entrée d'une seule donnée. Le bit 0 de l'octet d'adresse 4021 $_h$ sert de discriminateur. Il vaut 1 dans le premier cas et 0 dans le second.

13. Nombre d'octets libres

Les octets restant libres dans le système se situent entre les adresses pointées par :
— STKEND : fin de la pile du calculateur, évoluant vers le haut de la mémoire et :
— SP : pointeur de la pile de la machine, progressant vers le bas.

La fonction décrite évalue la différence (SP) - (STKEND) et affiche le résultat en décimal.

Fonctionnement

Il suffit de faire RAND USR 31000 et « 0 ».

Si la commande « X » est en cours, presser N/L pour continuer.

Description

Étant donné qu'il n'existe pas d'instruction de transfert direct de SP à HL, on copie SP en mémoire afin de lire son contenu. DE reçoit la valeur de STKEND. Lorsque la soustraction a été effectuée, son résultat est passé dans BC. Un saut au

sous-programme affichant la valeur de BC en décimal achève le travail.

Module commun utilisé

7DFC $_h$: Inscrit la valeur décimale de BC à l'écran.

Somme de contrôle

De 7C18 $_h$ à 7C29 $_h$ (31768-31785) : 2224.

14. Conversion en décimal

Ce module donne, pour les valeurs hexadécimales comprises entre 1 et FFFF $_h$, l'équivalent décimal.

Fonctionnement

Frapper RAND USR 31000 puis « D ». Répondre au point d'interrogation en indiquant la valeur à convertir (les 4 derniers chiffres hexa étant pris en compte). Terminer par un point.

Exemple : ?7BC2. = 31682.

La machine continue de réclamer des entrées jusqu'à ce qu'on presse un point seul (suivi de N/L si les fonctions multiples sont opérationnelles).

Description

Là encore, il s'agit de l'emploi de ressources préexistantes. Une boucle allant de 7C2A $_h$ à 7C2F $_h$ appelle la routine de conversion proprement dite, tant que la valeur introduite n'est pas nulle (point seul).

Le programme de conversion commence par un saut de ligne et l'inscription de " ?". Ensuite, la routine débutant en 7C41 $_h$ capte le nombre en hexa au clavier et le dispose dans BC.

Enfin, aussitôt après l'impression du signe "=", un saut en 7DFC $_h$ affiche la valeur décimale du contenu de BC.

Modules communs utilisés

7C41 $_h$: Entrée d'un nombre en hexa dans BC.
7DFC $_h$: Affichage du contenu de BC, en décimal.

Somme de contrôle

De 7C2A $_h$ à 7C40 $_h$ (31786-31808) : 3098.

15. Entrée d'une adresse en hexadécimal dans BC

En fait, il s'agit de l'entrée de 4 digits hexadécimaux représentant fréquemment une adresse.

Ce module n'est pas accessible par l'aiguillage (ce n'est pas une fonction à part entière) mais peut fort bien être appelé par RAND USR 31809.

Description

A priori le contenu de BC est annulé de sorte que, si l'on sort immédiatement de la routine, la valeur ramenée sera 0.

Puis le clavier est scruté et CALL 14D2 établit la retenue à 1 si et seulement si la touche pressée correspond à un chiffre ou une lettre.

Dans tous les autres cas (un point ou N/L par exemple), le retour s'effectue.

Ensuite, le code obtenu est comparé à celui de la lettre « G ». Il s'avérera supérieur ou égal pour toute lettre postérieure à F dans l'alphabet. Celle-ci ne signifiant rien en hexadécimal n'est pas prise en compte.

Une entrée correcte se verra par contre affichée à l'écran. Le petit utilitaire va transformer le code contenu dans l'accumulateur en 16 fois la valeur réelle de l'entrée.

Par exemple, si on a pressé la touche 2 (code $1E_h$), A contiendra 32 (20_h).

Un marqueur servant à compter les décalages et rotations qui vont suivre est introduit (OR 08). On procède alors à l'introduction dans BC de la valeur du digit hexa. Celle-ci s'achève lorsque, le marqueur passant dans la retenue, l'accumulateur s'annule. La prochaine entrée est ensuite considérée.

Modules communs utilisés

$7C61_h$: Utilitaire de décalage.
$7E81_h$: Scrutation du clavier.

Somme de contrôle

De $7C41_h$ à $7C6B_h$ (31809-31851) : 5614.
L'utilitaire est compris dans cette somme.

16. Titres des programmes sur cassette

Cette commande donne les titres des programmes qui se trouvent sur une cassette.

Fonctionnement

Faire RAND USR 31000 puis « K » (pour K7).

L'ordinateur se place en attente (comme pour LOAD) et

détecte les silences qui précèdent les programmes. Puis, il lit chaque titre, l'affiche et attend qu'on presse N/L pour continuer. Ne pas interrompre le défilement de la bande durant l'affichage.

S'il reste des « débris » entre les programmes, des titres fantaisistes sont affichés. La même chose survient si on débute la lecture au milieu d'un enregistrement.

On peut presser BREAK à tout moment.

Description

La compréhension de la méthode employée nécessite l'étude de la routine LOAD (chapitre 4-paragraphe B).

En effet, de 7C74 $_h$ à 7C88 $_h$ et de 7CA7 $_h$ à 7CC1 $_h$ se trouve le sous-programme de lecture des enregistrements identique en tout point à celui de la ROM.

Après commutation en mode FAST (CALL OF23), une boucle guette un silence sur la bande magnétique.

Puis, une seconde boucle (7C89 $_h$-7C95 $_h$) inscrit un à un sur l'écran les caractères lus. La fin du titre est détectée grâce à l'élément binaire de poids fort (bit 7) de la dernière lettre (cf. chapitre V).

La machine retourne alors en mode SLOW et attend qu'une touche (par exemple N/L) soit pressée. Lorsque cela se produit, un passage à la ligne d'écran suivante s'opère avant la recherche d'un nouveau titre.

Sur un BREAK, un saut en 7D38 $_h$ provoque le retour au Basic, en mode SLOW.

Modules communs utilisés

7D38 $_h$: Sortie en mode SLOW.

Somme de contrôle

De 7C6C $_h$ à 7CC1 $_h$ (31852-31937) : 10469.

17. Contrôle des enregistrements sur cassette

Cette fonction permet de vérifier qu'un programme a été correctement copié sur cassette.

Elle agit exactement comme un LOAD (sans modifier le contenu de la machine, néanmoins), c'est-à-dire que la lecture commence aussitôt après qu'un silence (même très court) a été perçu sur la bande magnétique. Le nom du programme n'est pas contrôlé. Par conséquent, pour s'assurer de contrôler l'enregistrement X voulu, il faut :
— soit positionner la bande au niveau de la zone de silence produite par le ZX 81 avant tout programme, puis frapper RAND USR 31000 suivi de « C » et enclencher la lecture ;
— soit positionner la bande n'importe où au milieu de l'enregistrement Y précédant X, enclencher *d'abord* la lecture puis faire RAND USR 31000, « C ».

Cette seconde méthode s'avère plus confortable à l'usage.

Fonctionnement

Ainsi que nous venons de le voir, la lettre C appelle cette fonction.

La vérification consiste à comparer octet par octet le contenu de la cassette avec celui de la mémoire. Si une différence se manifeste, l'adresse de l'octet modifié est affichée en hexadécimal sur l'écran.

On peut faire BREAK à tout moment.

A la fin de l'exécution, si la commande « X » n'est pas activée, le retour au Basic a lieu en mode FAST.

Étant donné que les variables-système se trouvent également inscrites sur la bande magnétique et que ces variables sont affectées presque en permanence, voici une liste des modifications possibles qui, même si le contrôle est fait aussitôt après le « SAVE », ne signalent pas une erreur.

Les adresses en italique sont celles des octets systématiquement différents alors que les parenthèses indiquent une altération possible seulement :

- (400A-400B) et (402B-402C) sont parfois modifiés si l'on a refait la liste du programme Basic.
- *400E*-(400F)
- (4012-4013) altérés si l'on a sauvé après un CLEAR.
- (4016-4017)
- *401A*-(401B)
- *401C*-(401D)
- *401E*
- *4021*
- *4025-4026*
- (4027)
- *4030*-(4031)
- (4032-4033)
- *4034-4035*
- *4039-403A*
- *403B* si l'on a sauvé en mode SLOW.

En pratique, on constate qu'il est inutile de s'assurer que les différences relevées se trouvent bien dans dans cette liste car, si les connexions sont mauvaises, de très nombreux octets s'avéreront d'emblée altérés, de sorte que l'écran se remplit très vite d'adresses et qu'un retour s'effectue sur le compte rendu 5.

On vérifiera simplement que la liste des octets modifiés commence bien en $400E_h$ et s'achève en $403B_h$.

Notons enfin que, si le « SAVE » initial était effectué à partir de l'intérieur d'un programme Basic, on trouvera encore plusieurs différences dont la liste demeure imprévisible. Celles-ci se situent dans la zone des variables et dans l'espace de travail de l'ordinateur.

Description

Là encore, la lecture du chapitre 4-paragraphe B s'avère indispensable si l'on souhaite saisir le fonctionnement de cette routine. Nous retrouvons à nouveau la copie d'une

partie de la ROM (de 7CCE$_h$ à 7CE2$_h$ et de 7DID$_h$ à 7D37$_h$).

Le programme comporte, comme le précédent, une boucle de recherche de l'amorce silencieuse d'un enregistrement.

La survenue de cette dernière provoque :
— la première fois, le début du contrôle (après franchissement du titre) ;
— la seconde fois, le retour à la routine appelante.

Les deux situations sont distinguées par l'état d'un drapeau (BIT 7, (4021)).

Il peut arriver, assez exceptionnellement, qu'une coupure de modulation (« drop-out ») s'avère d'une durée suffisamment importante pour provoquer un retour alors que l'enregistrement n'a pas été entièrement vérifié et qu'aucun signal d'erreur n'est affiché. Lorsqu'on emploie des cassettes très bon marché, il convient donc de s'assurer, grâce au compteur du magnétophone ou bien à l'aide d'un chronomètre, que la durée du contrôle est bien égale à celle du « SAVE ».

Lorsqu'une erreur est décelée, l'adresse de l'octet incriminé est passée dans HL et affichée par le sous-programme spécialisé (en 7D3D$_h$).

Avant cela, on contrôle que cette adresse ne correspond pas à un emplacement de l'écran. En effet, ce dernier est effacé au moment du « SAVE ». Or, lors de la vérification, des résultats y sont inscrits. Par conséquent, la comparaison révélerait des dizaines de différences inutiles car non significatives.

BREAK a le même effet que pour la routine précédente.

Modules communs utilisés

7D38$_h$: Sortie en mode SLOW
7D3D$_h$: Affichage de HL.

Somme de contrôle

De 7CC2 $_h$ à 7D37 $_h$ (31938-32055) : 14367.

18. Sortie en mode SLOW

Cet utilitaire ultra-simple ne comporte qu'un appel à la routine SLOW et une sortie sur le compte rendu D.

Somme de contrôle

De 7D38 $_h$ à 7D3C $_h$ (32056-32060) : 482.

19. Affichage des registres HL et A

L'inscription du contenu de HL équivaut à passer H puis L dans l'accumulateur puis à invoquer le sous-programme dédié à l'affichage de A.

Cependant, si H s'avère nul, deux espaces le représenteront.

Pour écrire le contenu de A sur l'écran, on procède par moitié. Une série de décalages isole la partie gauche de l'octet tandis qu'un masquage (AND OF) sépare les 4 éléments binaires de droite.

Chaque moitié se voit ajouter 1C $_h$ puisque les codes des chiffres et des lettres partent de 1C $_h$ et non de zéro. RST 10 réalise l'affichage.

Somme de contrôle

De 7D3D $_h$ à 7D5B $_h$ (32061-32091) : 4190.

20. Liste des variables Basic

La commande V visualise la liste des variables Basic existantes.

Fonctionnement

Frapper RAND USR 31000 suivi de « V ».

La présentation sur l'écran est la suivante :

• pour les variables numériques simples ou dont le nom comporte plusieurs caractères :
nom = valeur
Exemples : A = 18
TOTO = 34 E8

• pour les variables de boucle :
nom = valeur > limite ; pas * ligne du FOR
Exemple : X = 6 > 10 ; 2 * 5.

• pour les variables de chaînes de caractères simples :
nom : longueur C (pour caractères)
Exemple : A$: 18 C

• pour les tableaux :
nom (dimension 1, dimension 2, etc.)
Exemples : A$ (5, 56, 2)
G (15, 3)

Si l'écran se remplit alors que toutes les variables n'ont pas été listées, le programme s'arrête jusqu'à ce que l'on presse :
— BREAK pour revenir au Basic,
— n'importe quelle autre touche pour faire apparaître la suite.

A la fin de l'exécution, si la fonction X est en cours, presser une touche pour continuer.

```
***EXEMPLE D"UTILISATION

S=8
I=13>22;1*40
K=0>1; -1*105
B$(10,4)
AD=32092
Z=201
FLAG=0
Y=201
P=1
U=1
W=8
T=40
V=5
O=11
L=0
AD2=2005
TA=37
Q=26032
WS=32073
T6=12
Y1=55
DRAP=1
```

Description

Il convient tout d'abord de rappeler comment les 3 éléments binaires les plus à gauche du nom d'une variable indiquent son type :

Bits 7	6	5	
0	1	1	Variable numérique simple
1	0	1	Variable numérique à nom long
0	1	0	Chaîne simple
1	1	1	Variable de boucle
1	0	0	Tableau de nombres
1	1	0	Tableau de chaînes

Par ailleurs, la valeur 80 $_h$ (= 1000 0000 binaire) signale la fin de la zone des variables.

Nous pouvons à présent analyser le progamme :

7D5C $_h$ - 7D63 $_h$: HL pointe sur la première variable. L'octet qui débute son nom est passé dans l'accumulateur. Le bit 7

ne vaut 0 que pour les chaînes et variables numériques simples.

7D64 $_h$ - 7D75 $_h$: cas d'une variable numérique simple. Son nom est affiché suivi du signe "=" et de sa valeur (imprimée par le sous-programme qui commence en 7E1F $_h$). CALL 7E07 fait passer à la ligne suivante et vérifie que le bas de l'écran n'est pas atteint.

7D76 $_h$ - 7D89 $_h$: cas d'une chaîne simple. Après le nom s'inscrivent les signes "$" et " :", puis la longueur de la chaîne (grâce à la routine débutant en 7DF8 $_h$) et la lettre "C". La somme (HL) + (BC) pointe sur la variable suivante.

7D8A $_h$ - 7D8C $_h$: si la liste des variables est achevée, (octet 80 $_h$), le retour s'effectue.

7D8D $_h$ - 7D92 $_h$: le bit 7 vaut 1 pour tous les types restants. Il n'a plus d'intérêt. Le test CP 60 permet d'isoler les variables de boucle.

7D93 $_h$ - 7DB4 $_h$: cas d'une variable de boucle. Tous les paramètres sont affichés avec les séparatifs appropriés.

7DB5 $_h$ - 7DB8 $_h$: parmi les types restants, les variables numériques à nom long sont les seules pour lesquelles le bit 5 vaut 1.

7DB9 $_h$ - 7DC4 $_h$: leur nom est affiché, le dernier caractère se distinguant par son élément binaire de poids fort (bit 7) valant 1. La suite demeure la même que pour une variable numérique simple.

7DC5 $_h$ - 7DCA $_h$: les tableaux de chaînes se caractérisent par le bit 6 valant 1.

7DCB $_h$ - 7DD3 $_h$: le nom du tableau s'inscrit, suivi de "$" et d'une parenthèse ouvrante.

7DD4 $_h$ - 7DDB $_h$: la longueur totale du tableau est passée dans BC puis sauvegardée sur la pile machine. B reçoit ensuite le nombre de dimensions.

7DDC $_h$ - 7DE5 $_h$: impression des dimensions séparées par des virgules.

7DE6 $_h$ - 7DF4 $_h$: la position d'écriture est ramenée d'un pas

en arrière afin d'effacer la dernière virgule inutile, remplacée par une parenthèse fermante. (voir chapitre 1-paragraphe B). Après ajout de la longueur totale du tableau (récupérée dans BC), HL pointera sur la variable suivante.

$7DF5_h$ - $7DF7_h$: cas d'un tableau de nombres. Le traitement reste le même que pour un tableau de chaînes mais le signe "$" ne doit pas exister après le nom.

$7DF8_h$ - $7E06_h$: routine affichant en décimal la valeur formée par deux octets successifs (longueur d'une chaîne, dimension d'un tableau ou numéro de ligne) pointés par HL (voir le chapitre sur le calculateur).

$7E07_h$ - $7E1E_h$: test contrôlant si le bas de l'écran est atteint. Dans ce cas, lecture du clavier et retour au Basic si BREAK (espace) est pressé. Sinon, effacement de l'écran et poursuite de l'exécution.

$7E1F_h$ - $7E30_h$: routine transférant un nombre sur la pile avant d'afficher sa valeur en décimal (voir le chapitre sur le calculateur).

Modules communs utilisés

$7DFC_h$: affichage du contenu du registre BC sur l'écran, en décimal.
$7E81_h$: scrutation du clavier.

Somme de contrôle

De $7D5C_h$ à $7E30_h$ (32092-32304) : 27352.

21. Entrées au clavier et scrutation du clavier

Cet utilitaire permet de saisir deux nombres en décimal au clavier. Les deux valeurs doivent être séparées par une virgule et achevées par un point.

Il est possible de ne prendre qu'un seul nombre en annulant le bit 0 de l'objet d'adresse 4021 $_h$ et en entrant en 7E35 $_h$.

Description

Un point d'interrogation est d'abord affiché.

Dans le but d'employer le convertisseur de données du Basic, la variable système CH-ADD se voit modifiée, sa valeur d'origine étant préservée en mémoire (voir le chapitre sur le calculateur).

Le clavier est alors scruté et des comparaisons (CP) permettent de rejeter les entrées invalides et de détecter les séparatifs (virgule et point). Les chiffres corrects sont disposés dans le tampon de l'imprimante.

A partir de 7E58 $_h$, ces chiffres vont être transformés avec l'aide de la ROM Basic en une valeur en virgule flottante résidant au sommet de la pile du calculateur.

Enfin, une ou deux valeurs (selon l'état du drapeau prévu à cet effet) seront dépilées vers le registre BC et placées dans les emplacements de mémoire d'adresses (403E $_h$ - 403Fh) et (4040 $_h$-4041 $_h$). CH-ADD est restaurée avant le retour.

La routine de scrutation du clavier ne nécessite aucun commentaire puisqu'elle est la reproduction exacte de celle élaborée au chapitre 1.

Sommes de contrôle

De 7E31 $_h$ à 7E80 $_h$ (32305-32384) : 8765.
De 7E81 $_h$ à 7E92 $_h$ (32385-32402) : 2311.

22. Rénuméroteur de lignes Basic

Ce sous-programme a déjà été décrit dans la revue *Micro-Systèmes* (numéro 35, octobre 1983).

Fonctionnement

Faire RAND USR 31000 puis « R ».

En réponse au point d'interrogation, donner le numéro que devra avoir la première ligne dans la nouvelle numérotation, puis le pas de celle-ci.

Exemple : ?5,10.
renumérotera à partir de 5 par pas de 10.

Si un GOTO ou un GOSUB renvoie à une ligne de numéro supérieur au plus haut existant, la renumérotation n'aura pas lieu. Dans le cas contraire, les arguments de ces instructions se verront automatiquement adaptés à la nouvelle numérotation.

Normalement, l'ensemble de la liste Basic est modifiée. Par contre si le programme comporte un STOP, la renumérotation s'arrêtera là.

Durant l'exécution, il est possible que l'écran clignote. A la fin de celle-ci, si la commande « X » est activée, presser N/L pour continuer.

Description

Nous invitons le lecteur à consulter la revue citée plus haut pour une description complète.

Par rapport à la version publiée antérieurement, deux légères modifications ont été apportées :

1) Au début du programme, quelques instructions font appel à la routine d'entrée de deux valeurs décimales au clavier afin d'introduire aisément les paramètres de la renumérotation.

2) En 7EE6, on a ajouté un appel de sous-programme qui se substitue aux instructions : CP DE
DEC DE
INC HL

Ceci vise à éliminer une possibilité d'erreur rarissime. En effet, si le codage Sinclair d'un nombre comprenait les octets DE EC successivement, la version précédente, croyant avoir affaire à un THEN GOTO, tentait de traiter l'argument d'un saut imaginaire. Cela survenait par exemple lors de la rénumérotation de la ligne suivante : 10 IF × = 14606060 THEN PRINT.

Pour l'éviter, les codages Sinclair sont ignorés dans cette version.

Modules communs utilisés

7E31 $_h$: Entrée de deux nombres décimaux au clavier.

7FE0 $_h$: Transfert dans le registre DE de la longueur d'une ligne Basic et, dans l'accumulateur, du code de la première instruction de cette ligne.

Somme de contrôle

De 7E93 $_h$ à 7FFE $_h$ (32403-32766) : 40347.

3

Programmes

DESASSEMBLEUR

```
   1 REM
   2 LET D$="BCDEHL+A"
   5 LET AD=0
  10 PRINT "ADRESSE?"
  15 INPUT X$
  20 IF X$="S" THEN LET X$=STR$
 AD
  25 IF X$(LEN X$)="H" THEN GOSU
B 605
  30 LET AD=VAL X$
  35 CLS
  40 FOR I=1 TO 21
  45 LET AD2=AD
  50 GOSUB 245
  55 PRINT V$;TAB 5;
  60 LET M$=""
  65 LET Z=PEEK AD
  70 GOSUB 215
  75 LET FLAG=0
  80 IF Z=203 THEN GOSUB 275
  85 IF Z=221 THEN GOSUB 385
  90 IF Z=237 THEN GOSUB 555
  95 IF Z=253 THEN GOSUB 395
 100 IF M$="" THEN LET M$=A$(Z+1
)
 105 FOR K=LEN M$ TO 1 STEP -1
 110 IF M$(K)<>"$" THEN GOTO 170
 115 GOSUB 210
 120 IF K=LEN M$ THEN LET M$=M$(
TO K-1)+V$
 125 IF K<LEN M$ THEN LET M$=M$(
TO K-1)+V$+M$(K+1 TO )
 130 NEXT K
 135 IF PEEK 16441<17 THEN GOTO
150
 140 PRINT ".";
 145 GOTO 135
 150 PRINT M$
 155 LET AD=AD+1
 160 NEXT I
 165 GOTO 10
 170 IF M$(K)<>"£" THEN GOTO 130
 175 GOSUB 210
 180 IF Y>127 THEN GOTO 200
```

```
185 LET AD2=AD+Y+1
190 GOSUB 245
195 GOTO 120
200 LET AD2=AD-255+Y
205 GOTO 190
210 LET AD=AD+1
215 LET Y=PEEK AD
220 LET TB=INT (Y/16)
225 LET TA=Y-TB*16+28
230 LET V$=CHR$ (TB+28)+CHR$ TA
235 PRINT V$;".";
240 RETURN
245 LET V$=""
250 LET T=AD2-INT (AD2/16)*16+2
8
255 LET V$=CHR$ T+V$
260 LET AD2=INT (AD2/16)
265 IF NOT AD2 THEN RETURN
270 GOTO 250
275 GOSUB 210
280 LET Y1=Y
285 IF Y1>=64 THEN GOTO 360

290 IF Y1>47 AND Y1<56 THEN GOT
O 495
295 IF Y1>=56 THEN LET Y1=Y1-8
300 LET Y=INT (Y1/8)+1
305 LET M$=B$(Y)
310 LET W=Y1-(Y-1)*8+1
315 IF W=7 THEN GOTO 335
320 IF FLAG THEN GOTO 495
325 LET M$=M$+" "+D$(W)
330 RETURN
335 IF FLAG THEN GOTO 350
340 LET M$=M$+"(HL)"
345 RETURN
350 LET M$=M$+"("+H$+")"
355 RETURN
360 LET Y=INT (PEEK AD/64)
365 LET L=PEEK AD-Y*64
370 LET M$=B$(Y+7)+STR$ INT (L/
8)+","
375 LET W=L-INT (L/8)*8+1
380 GOTO 315
385 LET H$="IX"
```

suite au verso

```
390 GOTO 400
395 LET H$="IY"
400 LET FLAG=1
405 GOSUB 210
410 LET DRAP=0
415 IF Y<>203 THEN GOTO 430
420 LET DRAP=1
425 GOTO 535
430 LET U=0
435 FOR K=1 TO 14
440 IF Y<>F(K) THEN GOTO 455
445 LET U=1
450 LET K=14
455 NEXT K
460 IF U THEN GOTO 505
465 FOR K=1 TO 25
470 IF Y<>G(K) THEN GOTO 485
475 LET U=1
480 LET K=25
485 NEXT K
490 IF U THEN GOTO 530
495 LET M$="???"
500 RETURN
505 LET M$=A$(Y+1)
510 FOR K=1 TO LEN M$-1
515 IF M$(K TO K+1)="HL" THEN LET M$=M$( TO K-1)+H$+M$(K+2 TO )
520 NEXT K
525 RETURN
530 LET M$=A$(Y+1)
535 GOSUB 210
540 LET H$=H$+"+"+V$
545 IF NOT DRAP THEN GOTO 510
550 GOTO 275
555 GOSUB 210
560 LET P=0
565 FOR K=1 TO 58
570 IF VAL (C$(K, TO 3))<>Y THEN GOTO 590
575 LET M$=C$(K,4 TO )
580 LET K=58
585 LET P=1
590 NEXT K
595 IF P THEN RETURN
```

```
600 GOTO 495
605 LET WS=0
610 FOR K=1 TO LEN X$-1
615 LET WS=WS*16+CODE X$(K)-28
620 NEXT K
625 LET X$=STR$ WS
630 RETURN
635 SAVE "HEXA-DES[illegible]"
640 GOTO 1
```

FONCTIONS MULTIPLES

```
78FC 21.FC.78....LD HL,78FC
78FF E5..........PUSH HL
7900 CD.81.7E....CALL 7E81
7903 CD.2A.0A....CALL 0A2A
7906 18.10.......JR 7918
```

RETOUR AU BASIC

```
7908 CF..........RST 8
7909 0D..........DEC C
```

TABLE DES COMMANDES

```
790A 39...........ADD HL,SP
790B 38.32.......JR C,793F
790D 2C...........INC L
790E 2A.2D.34....LD HL,(342D)
7911 29...........ADD HL,HL
7912 30.28.......JR NC,793C
7914 3B...........DEC SP
7915 37...........SCF
7916 27...........DAA
7917 3D...........DEC A
```

AIGUILLAGE

```
7918 CD.2B.0F....CALL 0F2B
791B 3E.12.......LD A,12
791D D7..........RST 10
791E CD.81.7E....CALL 7E81
7921 21.0A.79....LD HL,790A
7924 16.00.......LD D,00
7926 1E.0E.......LD E,0E
7928 BE..........CP (HL)
7929 28.06.......JR Z,7931
792B 23..........INC HL
792C 1D..........DEC E
792D 20.F9.......JR NZ,7928
792F 18.ED.......JR 791E
7931 D7..........RST 10
7932 3E.76.......LD A,76
7934 D7..........RST 10
7935 21.44.79....LD HL,7944
7938 3E.0E.......LD A,0E
793A 93..........SUB E
793B 5F..........LD E,A
793C CB.23.......SLA E
793E 19..........ADD HL,DE
793F 56..........LD D,(HL)
7940 23..........INC HL
7941 5E..........LD E,(HL)
7942 EB..........EX DE,HL
7943 E9..........JP (HL)
```

TABLE DES ADRESSES

```
7944  79...........LD  A,C
7945  60...........LD  H,B
7946  79...........LD  A,C
7947  7D...........LD  A,L
7948  79...........LD  A,C
7949  B6...........OR  (HL)
794A  7B...........LD  A,E
794B  C2.7B.E4.....JP  NZ,E47B
794E  7B...........LD  A,E
794F  FE.7C........CP  7C
7951  18.7C........JR  79CF
7953  2A.7C.6C.....LD  HL,(6C7C)
7956  7C...........LD  A,H
7957  C2.7D.5C.....JP  NZ,5C7D
795A  7E...........LD  A,(HL)
795B  93...........SUB E
795C  79...........LD  A,C
795D  08...........EX  AF,AF*
795E  78...........LD  A,B
795F  FC.
```

TRANSFERT EN REM

```
7960 CD.31.7E....CALL 7E31
7963 2A.40.40....LD HL,(4040)
7966 ED.5B.3E.40.LD DE,(403E)
796A D5..........PUSH DE
796B A7..........AND A
796C ED.52.......SBC HL,DE
796E 23..........INC HL
796F 44..........LD B,H
7970 4D..........LD C,L
7971 C5..........PUSH BC
7972 CD.C5.7B....CALL 7BC5
7975 C1..........POP BC
7976 E1..........POP HL
7977 11.82.40....LD DE,4082
797A ED.B0.......LDIR
797C C9..........RET
```

SUPPRESSION DES REM

```
797D CD.18.7C....CALL 7C18
7980 3E.12.......LD A,12
7982 D7..........RST 10
7983 21.7D.40....LD HL,407D
7986 7E..........LD A,(HL)
7987 FE.76.......CP 76
7989 CA.18.7C....JP Z,7C18
798C 23..........INC HL
798D CD.E0.7F....CALL 7FE0
7990 FE.EA.......CP EA
7992 EB..........EX DE,HL
7993 19..........ADD HL,DE
7994 20.F0.......JR NZ,7986
7996 1B..........DEC DE
7997 1B..........DEC DE
7998 1B..........DEC DE
7999 1B..........DEC DE
799A CD.F5.7B....CALL 7BF5
799D 18.E4.......JR 7983
```

MONITEUR HEXADECIMAL

```
799F  3E.17.......LD A,17
79A1  D7..........RST 10
79A2  C3.41.7C....JP 7C41
79A5  01.00.14....LD BC,1400
79A8  C3.F5.08....JP 08F5
79AB  CD.2A.0A....CALL 0A2A
79AE  3E.76.......LD A,76
79B0  D7..........RST 10
79B1  2A.7B.40....LD HL,(407B)
79B4  18.0D.......JR 79C3
79B6  CD.2A.0A....CALL 0A2A
79B9  CD.9F.79....CALL 799F
79BC  FE.76.......CP 76
79BE  20.F9.......JR NZ,79B9
79C0  D7..........RST 10
79C1  60..........LD H,B
79C2  69..........LD L,C
79C3  22.7B.40....LD (407B),HL
79C6  22.40.40....LD (4040),HL
79C9  06.14.......LD B,14
79CB  CD.3D.7D....CALL 7D3D
79CE  3E.16.......LD A,16

79D0  D7..........RST 10
79D1  0E.08.......LD C,08
79D3  7E..........LD A,(HL)
79D4  CD.49.7D....CALL 7D49
79D7  AF..........XOR A
79D8  D7..........RST 10
79D9  23..........INC HL
79DA  0D..........DEC C
79DB  20.F6.......JR NZ,79D3
79DD  D7..........RST 10
79DE  D7..........RST 10
79DF  D7..........RST 10
79E0  10.E9.......DJNZ 79CB
79E2  01.05.01....LD BC,0105
79E5  ED.43.3C.40.LD (403C),BC
79E9  CD.F5.08....CALL 08F5
79EC  7E..........LD A,(HL)
79ED  F6.80.......OR 80
79EF  D7..........RST 10
79F0  ED.4B.39.40.LD BC,(4039)
79F4  C5..........PUSH BC
79F5  CD.B6.7B....CALL 7BB6
```

```
79F8 C1...........POP BC
79F9 0C...........INC C
79FA CD.18.09....CALL 0918
79FD CD.81.7E....CALL 7E81
7A00 A7...........AND A
7A01 28.B3........JR Z,79B6
7A03 FE.36........CP 36
7A05 C8...........RET Z
7A06 CD.D2.14....CALL 14D2
7A09 30.39........JR NC,7A44
7A0B FE.2C........CP 2C
7A0D 30.EE........JR NC,79FD
7A0F D7...........RST 10
7A10 CD.61.7C....CALL 7C61
7A13 F5...........PUSH AF
7A14 CD.81.7E....CALL 7E81
7A17 CD.D2.14....CALL 14D2
7A1A 30.F8........JR NC,7A14
7A1C FE.2C........CP 2C
7A1E 30.F4........JR NC,7A14
7A20 D7...........RST 10
7A21 D6.1C........SUB 1C

7A23 57...........LD D,A
7A24 F1...........POP AF
7A25 82...........ADD A,D
7A26 CD.AA.7B....CALL 7BAA
7A29 AF...........XOR A
7A2A D7...........RST 10
7A2B 3A.39.40....LD A,(4039)
7A2E FE.04........CP 04
7A30 2A.0E.40....LD HL,(400E)
7A33 20.B7........JR NZ,79EC
7A35 ED.4B.3C.40.LD BC,(403C)
7A39 04...........INC B
7A3A 78...........LD A,B
7A3B FE.15........CP 15
7A3D D2.AB.79....JP NC,79AB
7A40 0E.05........LD C,05
7A42 18.A1........JR 79E5
7A44 ED.4B.39.40.LD BC,(4039)
7A48 57...........LD D,A
7A49 2A.0E.40....LD HL,(400E)
7A4C 7E...........LD A,(HL)
7A4D E6.3F........AND 3F
```

suite au verso

```
7A4F  77..........LD  (HL),A
7A50  7A..........LD  A,D
7A51  FE.73.......CP  73
7A53  20.11.......JR  NZ,7A66
7A55  0D..........DEC  C
7A56  0D..........DEC  C
7A57  0D..........DEC  C
7A58  CD.AE.7B....CALL  7BAE
7A5B  79..........LD  A,C
7A5C  FE.05.......CP  05
7A5E  38.D5.......JR  C,7A35
7A60  CD.18.09....CALL  0918
7A63  C3.EC.79....JP  79EC
7A66  FE.72.......CP  72
7A68  20.1B.......JR  NZ,7A85
7A6A  0C..........INC  C
7A6B  0C..........INC  C
7A6C  0C..........INC  C
7A6D  11.FF.FF....LD  DE,FFFF
7A70  CD.A4.7B....CALL  7BA4
7A73  79..........LD  A,C
7A74  FE.1D.......CP  1D

7A76  38.E8.......JR  C,7A60
7A78  ED.4B.3C.40.LD  BC,(403C)
7A7C  05..........DEC  B
7A7D  0E.1A.......LD  C,1A
7A7F  C2.E5.79....JP  NZ,79E5
7A82  C3.AB.79....JP  79AB
7A85  FE.70.......CP  70
7A87  20.11.......JR  NZ,7A9A
7A89  04..........INC  B
7A8A  FD.35.3D....DEC  (IY+3D)
7A8D  11.F8.FF....LD  DE,FFF8
7A90  CD.A4.7B....CALL  7BA4
7A93  78..........LD  A,B
7A94  FE.18.......CP  18
7A96  30.EA.......JR  NC,7A82
7A98  18.C6.......JR  7A60
7A9A  FE.71.......CP  71
7A9C  20.11.......JR  NZ,7AAF
7A9E  05..........DEC  B
7A9F  FD.34.3D....INC  (IY+3D)
7AA2  11.08.00....LD  DE,0008
7AA5  CD.A4.7B....CALL  7BA4
```

```
7AA8 78...........LD A,B
7AA9 FE.04........CP 04
7AAB 38.D5........JR C,7A82
7AAD 18.E9........JR 7A98
7AAF FE.74........CP 74
7AB1 20.0C........JR NZ,7ABF
7AB3 CD.81.7E.....CALL 7E81
7AB6 CD.AA.7B.....CALL 7BAA
7AB9 CD.49.7D.....CALL 7D49
7ABC C3.29.7A.....JP 7A29
7ABF FE.77........CP 77
7AC1 20.39........JR NZ,7AFC
7AC3 FD.CB.21.7E.BIT 7,(IY+21)
7AC7 20.0D........JR NZ,7AD6
7AC9 FD.CB.21.FE.SET 7,(IY+21)
7ACD 2A.7B.40.....LD HL,(407B)
7AD0 22.3E.40.....LD (403E),HL
7AD3 C3.FD.79.....JP 79FD
7AD6 FD.CB.21.BE.RES 7,(IY+21)
7ADA ED.5B.7B.40.LD DE,(407B)
7ADE 2A.3E.40.....LD HL,(403E)
7AE1 37...........SCF

7AE2 ED.52........SBC HL,DE
7AE4 7C...........LD A,H
7AE5 3C...........INC A
7AE6 28.06........JR Z,7AEE
7AE8 FE.01........CP 01
7AEA 28.0A........JR Z,7AF6
7AEC 18.E5........JR 7AD3
7AEE CB.7D........BIT 7, L
7AF0 28.FA........JR Z,7AEC
7AF2 7D...........LD A,L
7AF3 C3.B6.7A.....JP 7AB6
7AF6 CB.7D........BIT 7, L
7AF8 28.F8........JR Z,7AF2
7AFA 18.F0........JR 7AEC
7AFC 2A.7B.40.....LD HL,(407B)
7AFF 3D...........DEC A
7B00 FE.78........CP 78
7B02 20.0A........JR NZ,7B0E
7B04 AF...........XOR A
7B05 CD.26.05.....CALL 0526
7B08 2A.7F.40.....LD HL,(407F)
7B0B 23...........INC HL
```

suite au verso

```
7B0C  18.05........JR 7B19
7B0E  FE.74........CP 74
7B10  20.16........JR NZ,7B28
7B12  CD.5C.05.....CALL 055C
7B15  2A.7F.40.....LD HL,(407F)
7B18  2B...........DEC HL
7B19  22.7F.40.....LD (407F),HL
7B1C  01.00.01.....LD BC,0100
7B1F  CD.F5.08.....CALL 08F5
7B22  2A.7B.40.....LD HL,(407B)
7B25  C3.C3.79.....JP 79C3
7B28  FE.75........CP 75
7B2A  20.23........JR NZ,7B4F
7B2C  ED.4B.39.40..LD BC,(4039)
7B30  C5...........PUSH BC
7B31  01.0A.00.....LD BC,000A
7B34  CD.F5.08.....CALL 08F5
7B37  2A.7B.40.....LD HL,(407B)
7B3A  7E...........LD A,(HL)
7B3B  37...........SCF
7B3C  17...........RLA
7B3D  F5...........PUSH AF

7B3E  3E.00........LD A,00
7B40  CE.1C........ADC A,1C
7B42  D7...........RST 10
7B43  F1...........POP AF
7B44  CB.27........SLA A
7B46  20.F5........JR NZ,7B3D
7B48  C1...........POP BC
7B49  CD.18.09.....CALL 0918
7B4C  C3.FD.79.....JP 79FD
7B4F  FE.DA........CP DA
7B51  20.3C........JR NZ,7B8F
7B53  ED.73.7B.40..LD (407B),SP
7B57  E5...........PUSH HL
7B58  2A.7B.40.....LD HL,(407B)
7B5B  E5...........PUSH HL
7B5C  F5...........PUSH AF
7B5D  C5...........PUSH BC
7B5E  D5...........PUSH DE
7B5F  CD.A5.79.....CALL 79A5
7B62  18.0A........JR 7B6E
7B64  29...........ADD HL,HL
7B65  2A.27.2B.....LD HL,(2B27)
```

```
7B68  26.28........LD H,28
7B6A  38.35........JR C,7BA1
7B6C  2D...........DEC L
7B6D  31.
7B6E  21.64.7B.....LD HL,7B64
7B71  06.05........LD B,05
7B73  7E...........LD A,(HL)
7B74  D7...........RST 10
7B75  23...........INC HL
7B76  7E...........LD A,(HL)
7B77  D7...........RST 10
7B78  23...........INC HL
7B79  3E.14........LD A,14
7B7B  D7...........RST 10
7B7C  D1...........POP DE
7B7D  E5...........PUSH HL
7B7E  EB...........EX DE,HL
7B7F  C5...........PUSH BC
7B80  CD.3D.7D.....CALL 7D3D
7B83  C1...........POP BC
7B84  E1...........POP HL
7B85  AF...........XOR A
7B86  D7...........RST 10
7B87  10.EA........DJNZ 7B73
7B89  CD.81.7E.....CALL 7E81
7B8C  C3.B6.79.....JP 79B6


7B8F  FE.E1........CP E1
7B91  C2.FD.79.....JP NZ,79FD
7B94  2A.7B.40.....LD HL,(407B)
7B97  22.9E.7B.....LD (7B9E),HL
7B9A  CD.2A.0A.....CALL 0A2A
7B9D  CD.82.40.....CALL 4082
7BA0  18.B1........JR 7B53
7BA2  00...........NOP
7BA3  00...........NOP
7BA4  2A.7B.40.....LD HL,(407B)
7BA7  19...........ADD HL,DE
7BA8  18.08........JR 7BB2
7BAA  2A.7B.40.....LD HL,(407B)
7BAD  77...........LD (HL),A
7BAE  2A.7B.40.....LD HL,(407B)
7BB1  23...........INC HL
7BB2  22.7B.40.....LD (407B),HL
7BB5  C9...........RET
7BB6  01.18.00.....LD BC,0018
7BB9  CD.F5.08.....CALL 08F5
7BBC  2A.7B.40.....LD HL,(407B)
7BBF  C3.3D.7D.....JP 7D3D
```

GENERATEUR DE REM LONGUES

```
7BC2 CD.11.7C....CALL 7C11
7BC5 C5..........PUSH BC
7BC6 21.82.40....LD HL,4082
7BC9 CD.E7.02....CALL 02E7
7BCC CD.9E.09....CALL 099E
7BCF C1..........POP BC
7BD0 2A.7F.40....LD HL,(407F)
7BD3 09..........ADD HL,BC
7BD4 22.7F.40....LD (407F),HL
7BD7 EB..........EX DE,HL
7BD8 3E.16.......LD A,16
7BDA 77..........LD (HL),A
7BDB 2B..........DEC HL
7BDC 0B..........DEC BC
7BDD 78..........LD A,B
7BDE B1..........OR C
7BDF 20.F7.......JR NZ,7BD8
7BE1 C3.07.02....JP 0207
```

SUPPRESSION DE LIGNES

```
7BE4 CD.31.7E....CALL 7E31
7BE7 2A.3E.40....LD HL,(403E)
7BEA CD.D8.09....CALL 09D8
7BED E5..........PUSH HL
7BEE 2A.40.40....LD HL,(4040)
7BF1 CD.D8.09....CALL 09D8
7BF4 D1..........POP DE
7BF5 CD.E7.02....CALL 02E7
7BF8 CD.5D.0A....CALL 0A5D
7BFB C3.07.02....JP 0207
```

CONVERSION DECIMAL-->HEXA

```
7BFE CD.11.7C....CALL 7C11
7C01 60..........LD H,B
7C02 69..........LD L,C
7C03 3E.14.......LD A,14
7C05 D7..........RST 10
7C06 CD.3D.7D....CALL 7D3D
7C09 3E.76.......LD A,76
7C0B D7..........RST 10
7C0C 78..........LD A,B
7C0D B1..........OR C
7C0E C8..........RET Z
7C0F 18.ED.......JR 7BFE
```

ENTREE D"UNE SEULE DONNEE DE CIMALE

```
7C11 FD.CB.21.86.RES 0,(IY+21)
7C15 C3.35.7E....JP 7E35
```

NOMBRE D"OCTETS LIBRES

```
7C18 ED.73.7B.40.LD (407B),SP
7C1C 2A.7B.40....LD HL,(407B)
7C1F ED.5B.1C.40.LD DE,(401C)
7C23 ED.52.......SBC HL,DE
7C25 44..........LD B,H
7C26 4D..........LD C,L
7C27 C3.FC.7D....JP 7DFC
```

CONVERSION HEXA-->DECIMAL

```
7C2A CD.35.7C....CALL 7C35
7C2D 78..........LD A,B
7C2E B1..........OR C
7C2F C8..........RET Z
7C30 3E.76.......LD A,76
7C32 D7..........RST 10
7C33 18.F5.......JR 7C2A
7C35 3E.0F.......LD A,0F
7C37 D7..........RST 10
7C38 CD.41.7C....CALL 7C41
7C3B 3E.14.......LD A,14
7C3D D7..........RST 10
7C3E C3.FC.7D....JP 7DFC
```

ENTREE D''UNE ADRESSE EN HEXA DANS BC

```
7C41 01.00.00....LD BC,0000
7C44 C5..........PUSH BC
7C45 CD.81.7E....CALL 7E81
7C48 CD.D2.14....CALL 14D2
7C4B C1..........POP BC
7C4C D0..........RET NC
7C4D FE.2C.......CP 2C
7C4F 30.F4.......JR NC,7C45
7C51 D7..........RST 10
7C52 CD.61.7C....CALL 7C61
7C55 F6.08.......OR 08
7C57 CB.27.......SLA  A
7C59 28.E9.......JR Z,7C44
7C5B CB.11.......RL   C
7C5D CB.10.......RL   B
7C5F 18.F6.......JR 7C57
```

PETIT UTILITAIRE

```
7C61 D6.1C.......SUB 1C
7C63 CB.27.......SLA  A
7C65 CB.27.......SLA  A
7C67 CB.27.......SLA  A
7C69 CB.27.......SLA  A
7C6B C9..........RET
```

TITRES DES PROGRAMMES

```
7C6C  CD.23.0F....CALL 0F23
7C6F  CD.74.7C....CALL 7C74
7C72  18.FB.......JR 7C6F
7C74  0E.01.......LD C,01
7C76  06.00.......LD B,00
7C78  3E.7F.......LD A,7F
7C7A  DB.FE.......IN A,FE
7C7C  D3.FF.......OUT FF,A
7C7E  1F..........RRA
7C7F  D2.38.7D....JP NC,7D38
7C82  17..........RLA
7C83  17..........RLA
7C84  38.21.......JR C,7CA7
7C86  10.F0.......DJNZ 7C78
7C88  F1..........POP AF
7C89  CD.74.7C....CALL 7C74
7C8C  79..........LD A,C
7C8D  CB.7F.......BIT 7, A
7C8F  F5..........PUSH AF
7C90  E6.3F.......AND 3F
7C92  D7..........RST 10
7C93  F1..........POP AF

7C94  28.F3.......JR Z,7C89
7C96  CD.2B.0F....CALL 0F2B
7C99  FD.CB.3B.46.BIT 0,(IY+3B)
7C9D  28.FA.......JR Z,7C99
7C9F  CD.46.0F....CALL 0F46
7CA2  3E.76.......LD A,76
7CA4  D7..........RST 10
7CA5  18.C5.......JR 7C6C
7CA7  1E.94.......LD E,94
7CA9  06.1A.......LD B,1A
7CAB  1D..........DEC E
7CAC  DB.FE.......IN A,FE
7CAE  17..........RLA
7CAF  CB.7B.......BIT 7, E
7CB1  7B..........LD A,E
7CB2  38.F5.......JR C,7CA9
7CB4  10.F5.......DJNZ 7CAB
7CB6  20.04.......JR NZ,7CBC
7CB8  FE.56.......CP 56
7CBA  30.BA.......JR NC,7C76
7CBC  3F..........CCF
7CBD  CB.11.......RL   C
```

```
7CBF 30.B5........JR NC,7C76
7CC1 C9...........RET
```

VERIFICATION DES ENREGISTREMENTS

```
7CC2  CD.23.0F....CALL 0F23
7CC5  FD.CB.21.BE.RES 7,(IY+21)
7CC9  CD.CE.7C....CALL 7CCE
7CCC  18.FB.......JR 7CC9
7CCE  0E.01.......LD C,01
7CD0  06.00.......LD B,00
7CD2  3E.7F.......LD A,7F
7CD4  DB.FE.......IN A,FE
7CD6  D3.FF.......OUT FF,A
7CD8  1F..........RRA
7CD9  D2.38.7D....JP NC,7D38
7CDC  17..........RLA
7CDD  17..........RLA
7CDE  38.3D.......JR C,7D1D
7CE0  10.F0.......DJNZ 7CD2
7CE2  F1..........POP AF
7CE3  FD.CB.21.7E.BIT 7,(IY+21)
7CE7  C0..........RET NZ
7CE8  CD.CE.7C....CALL 7CCE
7CEB  CB.11.......RL  C
7CED  30.F9.......JR NC,7CE8
7CEF  21.09.40....LD HL,4009

7CF2  FD.CB.21.FE.SET 7,(IY+21)
7CF6  CD.CE.7C....CALL 7CCE
7CF9  79..........LD A,C
7CFA  BE..........CP (HL)
7CFB  23..........INC HL
7CFC  28.F8.......JR Z,7CF6
7CFE  E5..........PUSH HL
7CFF  2B..........DEC HL
7D00  EB..........EX DE,HL
7D01  2A.0C.40....LD HL,(400C)
7D04  A7..........AND A
7D05  ED.52.......SBC HL,DE
7D07  38.09.......JR C,7D12
7D09  EB..........EX DE,HL
7D0A  CD.3D.7D....CALL 7D3D
7D0D  3E.16.......LD A,16
7D0F  D7..........RST 10
7D10  18.08.......JR 7D1A
7D12  2A.10.40....LD HL,(4010)
7D15  A7..........AND A
7D16  ED.52.......SBC HL,DE
7D18  38.EF.......JR C,7D09
```

```
7D1A  E1............POP  HL
7D1B  18.D9.........JR  7CF6
7D1D  1E.94.........LD  E,94
7D1F  06.1A.........LD  B,1A
7D21  1D............DEC  E
7D22  DB.FE.........IN  A,FE
7D24  17............RLA
7D25  CB.7B.........BIT  7,  E
7D27  7B............LD  A,E
7D28  38.F5.........JR  C,7D1F
7D2A  10.F5.........DJNZ  7D21
7D2C  20.04.........JR  NZ,7D32
7D2E  FE.56.........CP  56
7D30  30.9E.........JR  NC,7CD0
7D32  3F............CCF
7D33  CB.11.........RL   C
7D35  30.99.........JR  NC,7CD0
7D37  C9............RET
```

SORTIE EN MODE SLOW

```
7D38 CD.2B.0F....CALL 0F2B
7D3B CF..........RST 8
7D3C 0C..........INC C
```

AFFICHAGE DE HL

```
7D3D 7C..........LD A,H
7D3E A7..........AND A
7D3F 20.04.......JR NZ,7D45
7D41 D7..........RST 10
7D42 D7..........RST 10
7D43 18.03.......JR 7D48
7D45 CD.49.7D....CALL 7D49
7D48 7D..........LD A,L
```

AFFICHAGE DE A

```
7D49 F5..........PUSH AF
7D4A CB.3F.......SRL  A
7D4C CB.3F.......SRL  A
7D4E CB.3F.......SRL  A
7D50 CB.3F.......SRL  A
7D52 C6.1C.......ADD A,1C
7D54 D7..........RST 10
7D55 F1..........POP AF
7D56 E6.0F.......AND 0F
7D58 C6.1C.......ADD A,1C
7D5A D7..........RST 10
7D5B C9..........RET
```

LISTE DES VARIABLES

```
7D5C  2A.10.40....LD HL,(4010)
7D5F  7E..........LD A,(HL)
7D60  CB.7F.......BIT 7, A
7D62  20.26.......JR NZ,7D8A
7D64  CB.B7.......RES 6, A
7D66  CB.6F.......BIT 5, A
7D68  28.0C.......JR Z,7D76
7D6A  D7..........RST 10
7D6B  3E.14.......LD A,14
7D6D  D7..........RST 10
7D6E  CD.1F.7E....CALL 7E1F
7D71  CD.07.7E....CALL 7E07
7D74  18.E9.......JR 7D5F
7D76  CB.EF.......SET 5, A
7D78  D7..........RST 10
7D79  3E.0D.......LD A,0D
7D7B  D7..........RST 10
7D7C  3C..........INC A
7D7D  D7..........RST 10
7D7E  CD.F8.7D....CALL 7DF8
7D81  AF..........XOR A
7D82  D7..........RST 10

7D83  3E.28.......LD A,28
7D85  D7..........RST 10
7D86  23..........INC HL
7D87  09..........ADD HL,BC
7D88  18.E7.......JR 7D71
7D8A  FE.80.......CP 80
7D8C  C8..........RET Z
7D8D  CB.BF.......RES 7, A
7D8F  FE.60.......CP 60
7D91  38.22.......JR C,7DB5
7D93  CB.B7.......RES 6, A
7D95  D7..........RST 10
7D96  3E.14.......LD A,14
7D98  D7..........RST 10
7D99  CD.1F.7E....CALL 7E1F
7D9C  3E.12.......LD A,12
7D9E  D7..........RST 10
7D9F  CD.20.7E....CALL 7E20
7DA2  3E.19.......LD A,19
7DA4  D7..........RST 10
7DA5  CD.20.7E....CALL 7E20
7DA8  3E.17.......LD A,17
```

```
7DAA  D7...........RST 10
7DAB  4E...........LD C,(HL)
7DAC  23...........INC HL
7DAD  46...........LD B,(HL)
7DAE  0B...........DEC BC
7DAF  23...........INC HL
7DB0  CD.FC.7D.....CALL 7DFC
7DB3  18.BC........JR 7D71
7DB5  CB.6F........BIT 5, A
7DB7  28.0C........JR Z,7DC5
7DB9  D7...........RST 10
7DBA  23...........INC HL
7DBB  7E...........LD A,(HL)
7DBC  CB.7F........BIT 7, A
7DBE  28.F9........JR Z,7DB9
7DC0  CB.BF........RES 7, A
7DC2  D7...........RST 10
7DC3  18.A6........JR 7D6B
7DC5  CB.EF........SET 5, A
7DC7  CB.77........BIT 6, A
7DC9  28.2A........JR Z,7DF5
7DCB  CB.B7........RES 6, A

7DCD  D7...........RST 10
7DCE  3E.0D........LD A,0D
7DD0  D7...........RST 10
7DD1  3E.10........LD A,10
7DD3  D7...........RST 10
7DD4  23...........INC HL
7DD5  4E...........LD C,(HL)
7DD6  23...........INC HL
7DD7  46...........LD B,(HL)
7DD8  C5...........PUSH BC
7DD9  23...........INC HL
7DDA  E5...........PUSH HL
7DDB  46...........LD B,(HL)
7DDC  C5...........PUSH BC
7DDD  CD.F8.7D.....CALL 7DF8
7DE0  C1...........POP BC
7DE1  3E.1A........LD A,1A
7DE3  D7...........RST 10
7DE4  10.F6........DJNZ 7DDC
7DE6  ED.4B.39.40..LD BC,(4039)
7DEA  0C...........INC C
7DEB  CD.18.09.....CALL 0918
```

```
7DEE E1...........POP HL
7DEF C1...........POP BC
7DF0 3E.11........LD A,11
7DF2 D7...........RST 10
7DF3 18.92........JR 7D87
7DF5 D7...........RST 10
7DF6 18.D9........JR 7DD1
7DF8 23...........INC HL
7DF9 4E...........LD C,(HL)
7DFA 23...........INC HL
7DFB 46...........LD B,(HL)
7DFC C5...........PUSH BC
7DFD E5...........PUSH HL
7DFE CD.20.15.....CALL 1520
7E01 CD.DB.15.....CALL 15DB
7E04 E1...........POP HL
7E05 C1...........POP BC
7E06 C9...........RET
7E07 3E.76........LD A,76
7E09 D7...........RST 10
7E0A ED.4B.39.40..LD BC,(4039)
7E0E 3E.02........LD A,02

7E10 B8...........CP B
7E11 D8...........RET C
7E12 E5...........PUSH HL
7E13 CD.81.7E.....CALL 7E81
7E16 A7...........AND A
7E17 CA.A6.03.....JP Z,03A6
7E1A CD.2A.0A.....CALL 0A2A
7E1D E1...........POP HL
7E1E C9...........RET
7E1F 23...........INC HL
7E20 E5...........PUSH HL
7E21 EF...........RST 28
7E22 A0...........AND B
7E23 34...........INC (HL)
7E24 EB...........EX DE,HL
7E25 E1...........POP HL
7E26 01.05.00.....LD BC,0005
7E29 ED.B0........LDIR
7E2B E5...........PUSH HL
7E2C CD.D6.15.....CALL 15D6
7E2F E1...........POP HL
7E30 C9...........RET
```

suite au verso

ENTREES AU CLAVIER

```
7E31 FD.CB.21.C6.SET 0,(IY+21)
7E35 3E.0F.......LD A,0F
7E37 D7..........RST 10
7E38 2A.16.40....LD HL,(4016)
7E3B 22.3C.40....LD (403C),HL
7E3E 21.3E.40....LD HL,403E
7E41 22.16.40....LD (4016),HL
7E44 E5..........PUSH HL
7E45 CD.81.7E....CALL 7E81
7E48 FE.1A.......CP 1A
7E4A 38.F9.......JR C,7E45
7E4C FE.26.......CP 26
7E4E 30.F5.......JR NC,7E45
7E50 E1..........POP HL
7E51 77..........LD (HL),A
7E52 D7..........RST 10
7E53 23..........INC HL
7E54 FE.1B.......CP 1B
7E56 20.EC.......JR NZ,7E44
7E58 DF..........RST 18
7E59 CD.48.15....CALL 1548
7E5C 23..........INC HL

7E5D 7E..........LD A,(HL)
7E5E CD.D2.14....CALL 14D2
7E61 22.16.40....LD (4016),HL
7E64 38.F3.......JR C,7E59
7E66 CD.8A.15....CALL 158A
7E69 ED.43.40.40.LD (4040),BC
7E6D FD.CB.21.46.BIT 0,(IY+21)
7E71 28.07.......JR Z,7E7A
7E73 CD.8A.15....CALL 158A
7E76 ED.43.3E.40.LD (403E),BC
7E7A 2A.3C.40....LD HL,(403C)
7E7D 22.16.40....LD (4016),HL
7E80 C9..........RET
```

SCRUTATION DU CLAVIER

```
7E81 FD.CB.3B.46.BIT 0,(IY+3B)
7E85 28.FA.......JR Z,7E81
7E87 ED.4B.25.40.LD BC,(4025)
7E8B CD.4B.0F....CALL 0F4B
7E8E CD.BD.07....CALL 07BD
7E91 7E..........LD A,(HL)
7E92 C9..........RET
```

RENUMEROTEUR

```
7E93  CD.31.7E....CALL 7E31
7E96  2A.3E.40....LD HL,(403E)
7E99  22.7B.40....LD (407B),HL
7E9C  3A.40.40....LD A,(4040)
7E9F  32.21.40....LD (4021),A
7EA2  CD.CB.7E....CALL 7ECB
7EA5  21.7D.40....LD HL,407D
7EA8  7E..........LD A,(HL)
7EA9  FE.76.......CP 76
7EAB  C8..........RET Z
7EAC  3A.7C.40....LD A,(407C)
7EAF  77..........LD (HL),A
7EB0  23..........INC HL
7EB1  3A.7B.40....LD A,(407B)
7EB4  77..........LD (HL),A
7EB5  CD.E0.7F....CALL 7FE0
7EB8  C8..........RET Z
7EB9  19..........ADD HL,DE
7EBA  E5..........PUSH HL
7EBB  3A.21.40....LD A,(4021)
7EBE  5F..........LD E,A
7EBF  16.00.......LD D,00

7EC1  2A.7B.40....LD HL,(407B)
7EC4  19..........ADD HL,DE
7EC5  22.7B.40....LD (407B),HL
7EC8  E1..........POP HL
7EC9  18.DD.......JR 7EA8
7ECB  21.7D.40....LD HL,407D
7ECE  7E..........LD A,(HL)
7ECF  FE.76.......CP 76
7ED1  C8..........RET Z
7ED2  23..........INC HL
7ED3  CD.E0.7F....CALL 7FE0
7ED6  C8..........RET Z
7ED7  FE.EC.......CP EC
7ED9  28.18.......JR Z,7EF3
7EDB  FE.ED.......CP ED
7EDD  28.14.......JR Z,7EF3
7EDF  FE.FA.......CP FA
7EE1  28.03.......JR Z,7EE6
7EE3  19..........ADD HL,DE
7EE4  18.E8.......JR 7ECE
7EE6  CD.EC.7F....CALL 7FEC
7EE9  00..........NOP
```

```
7EEA  7E...........LD  A,(HL)
7EEB  28.EA........JR  Z,7ED7
7EED  FE.76........CP  76
7EEF  28.F2........JR  Z,7EE3
7EF1  18.F3........JR  7EE6
7EF3  D5...........PUSH  DE
7EF4  E5...........PUSH  HL
7EF5  3E.7E........LD  A,7E
7EF7  23...........INC  HL
7EF8  BE...........CP  (HL)
7EF9  20.FC........JR  NZ,7EF7
7EFB  23...........INC  HL
7EFC  11.00.00.....LD  DE,0000
7EFF  7E...........LD  A,(HL)
7F00  C6.80........ADD  A,80
7F02  47...........LD  B,A
7F03  23...........INC  HL
7F04  7E...........LD  A,(HL)
7F05  CB.FF........SET  7, A
7F07  0E.08........LD  C,08
7F09  CB.27........SLA   A
7F0B  CB.13........RL    E

7F0D  CB.12........RL    D
7F0F  05...........DEC  B
7F10  28.07........JR  Z,7F19
7F12  0D...........DEC  C
7F13  20.F4........JR  NZ,7F09
7F15  23...........INC  HL
7F16  7E...........LD  A,(HL)
7F17  18.EE........JR  7F07
7F19  21.7B.40.....LD  HL,407B
7F1C  4E...........LD  C,(HL)
7F1D  23...........INC  HL
7F1E  46...........LD  B,(HL)
7F1F  23...........INC  HL
7F20  7E...........LD  A,(HL)
7F21  23...........INC  HL
7F22  E5...........PUSH  HL
7F23  6E...........LD  L,(HL)
7F24  67...........LD  H,A
7F25  B7...........OR  A
7F26  ED.52........SBC  HL,DE
7F28  E1...........POP  HL
7F29  23...........INC  HL
```

suite au verso

```
7F2A  30.21.......JR NC,7F4D
7F2C  D5..........PUSH DE
7F2D  5E..........LD E,(HL)
7F2E  23..........INC HL
7F2F  56..........LD D,(HL)
7F30  23..........INC HL
7F31  19..........ADD HL,DE
7F32  E5..........PUSH HL
7F33  21.21.40....LD HL,4021
7F36  5E..........LD E,(HL)
7F37  16.00.......LD D,00
7F39  EB..........EX DE,HL
7F3A  09..........ADD HL,BC
7F3B  44..........LD B,H
7F3C  4D..........LD C,L
7F3D  E1..........POP HL
7F3E  E5..........PUSH HL
7F3F  ED.5B.0C.40.LD DE,(400C)
7F43  ED.52.......SBC HL,DE
7F45  E1..........POP HL
7F46  D1..........POP DE
7F47  38.D7.......JR C,7F20

7F49  E1..........POP HL
7F4A  D1..........POP DE
7F4B  C1..........POP BC
7F4C  C9..........RET
7F4D  50..........LD D,B
7F4E  59..........LD E,C
7F4F  ED.53.3E.40.LD (403E),DE
7F53  21.00.00....LD HL,0000
7F56  06.10.......LD B,10
7F58  CB.23.......SLA  E
7F5A  CB.12.......RL   D
7F5C  7D..........LD A,L
7F5D  8D..........ADC A,L
7F5E  27..........DAA
7F5F  6F..........LD L,A
7F60  7C..........LD A,H
7F61  8C..........ADC A,H
7F62  27..........DAA
7F63  67..........LD H,A
7F64  10.F2.......DJNZ 7F58
7F66  EB..........EX DE,HL
7F67  E1..........POP HL
```

```
7F68  E5...........PUSH HL
7F69  23...........INC HL
7F6A  01.02.00.....LD BC,0002
7F6D  7A...........LD A,D
7F6E  CB.3F........SRL  A
7F70  CB.3F........SRL  A
7F72  CB.3F........SRL  A
7F74  CB.3F........SRL  A
7F76  CD.B3.7F.....CALL 7FB3
7F79  7A...........LD A,D
7F7A  E6.0F........AND 0F
7F7C  CD.B3.7F.....CALL 7FB3
7F7F  53...........LD D,E
7F80  0D...........DEC C
7F81  20.EA........JR NZ,7F6D
7F83  23...........INC HL
7F84  E5...........PUSH HL
7F85  2A.3C.40.....LD HL,(403C)
7F88  7E...........LD A,(HL)
7F89  B7...........OR A
7F8A  80...........ADD A,B
7F8B  77...........LD (HL),A

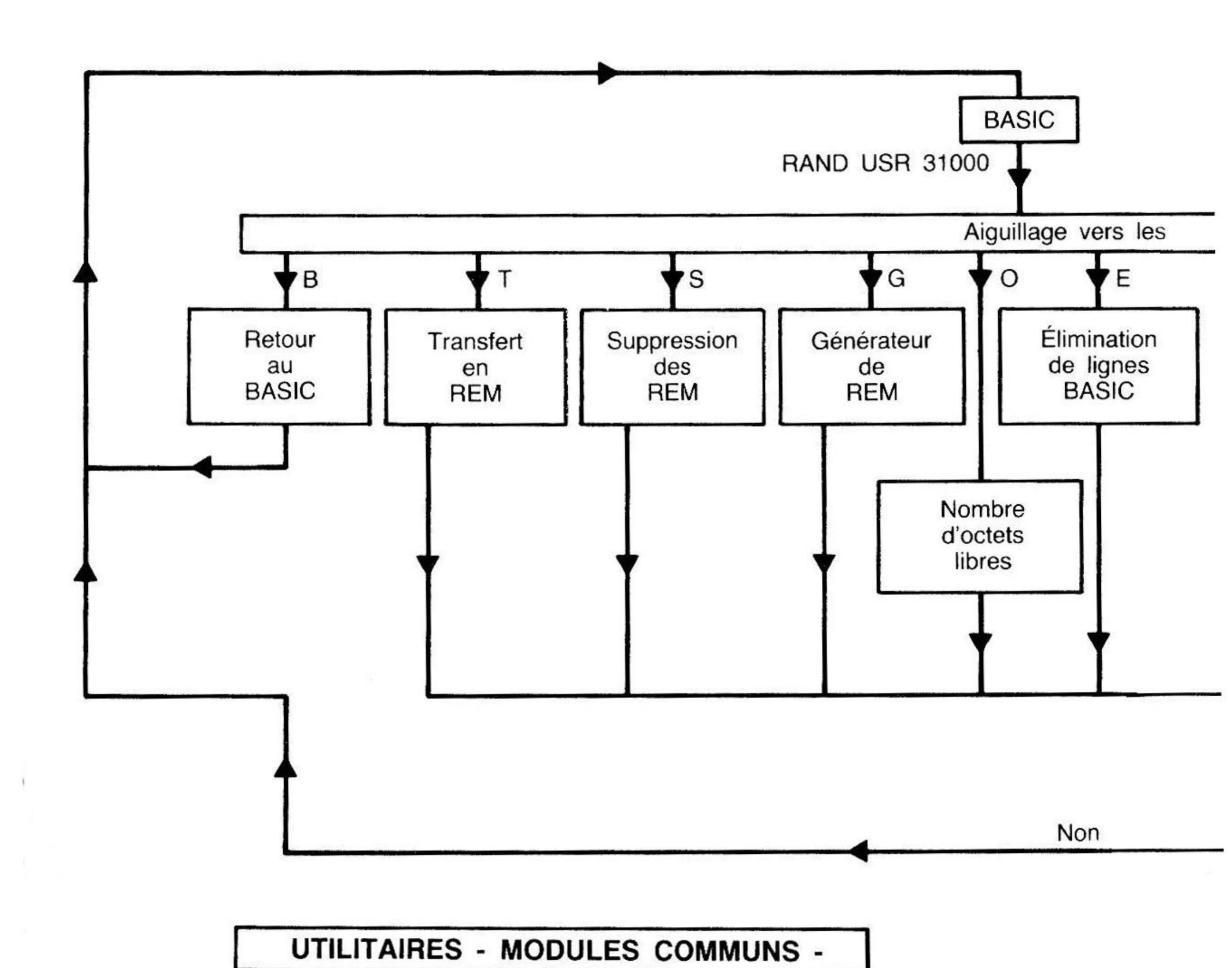
7F8C  23...........INC HL
7F8D  7E...........LD A,(HL)
7F8E  CE.00........ADC A,00
7F90  77...........LD (HL),A
7F91  E1...........POP HL
7F92  ED.5B.3E.40..LD DE,(403E)
7F96  B7...........OR A
7F97  06.10........LD B,10
7F99  CB.23........SLA  E
7F9B  CB.12........RL   D
7F9D  38.02........JR C,7FA1
7F9F  10.F8........DJNZ 7F99
7FA1  78...........LD A,B
7FA2  D6.80........SUB 80
7FA4  77...........LD (HL),A
7FA5  AF...........XOR A
7FA6  CB.3A........SRL  D
7FA8  CB.1B........RR   E
7FAA  23...........INC HL
7FAB  72...........LD (HL),D
7FAC  23...........INC HL
7FAD  73...........LD (HL),E
```

suite au verso

```
7FAE E1..........POP HL
7FAF D1..........POP DE
7FB0 C3.E3.7E....JP 7EE3
7FB3 C6.1C.......ADD A,1C
7FB5 32.46.40....LD (4046),A
7FB8 7E..........LD A,(HL)
7FB9 FE.7E.......CP 7E
7FBB 3A.46.40....LD A,(4046)
7FBE 20.1D.......JR NZ,7FDD
7FC0 ED.43.42.40.LD (4042),BC
7FC4 ED.53.44.40.LD (4044),DE
7FC8 CD.9B.09....CALL 099B
7FCB 23..........INC HL
7FCC F1..........POP AF
7FCD D1..........POP DE
7FCE 13..........INC DE
7FCF D5..........PUSH DE
7FD0 F5..........PUSH AF
7FD1 ED.4B.42.40.LD BC,(4042)
7FD5 ED.5B.44.40.LD DE,(4044)
7FD9 3A.46.40....LD A,(4046)
7FDC 04..........INC B

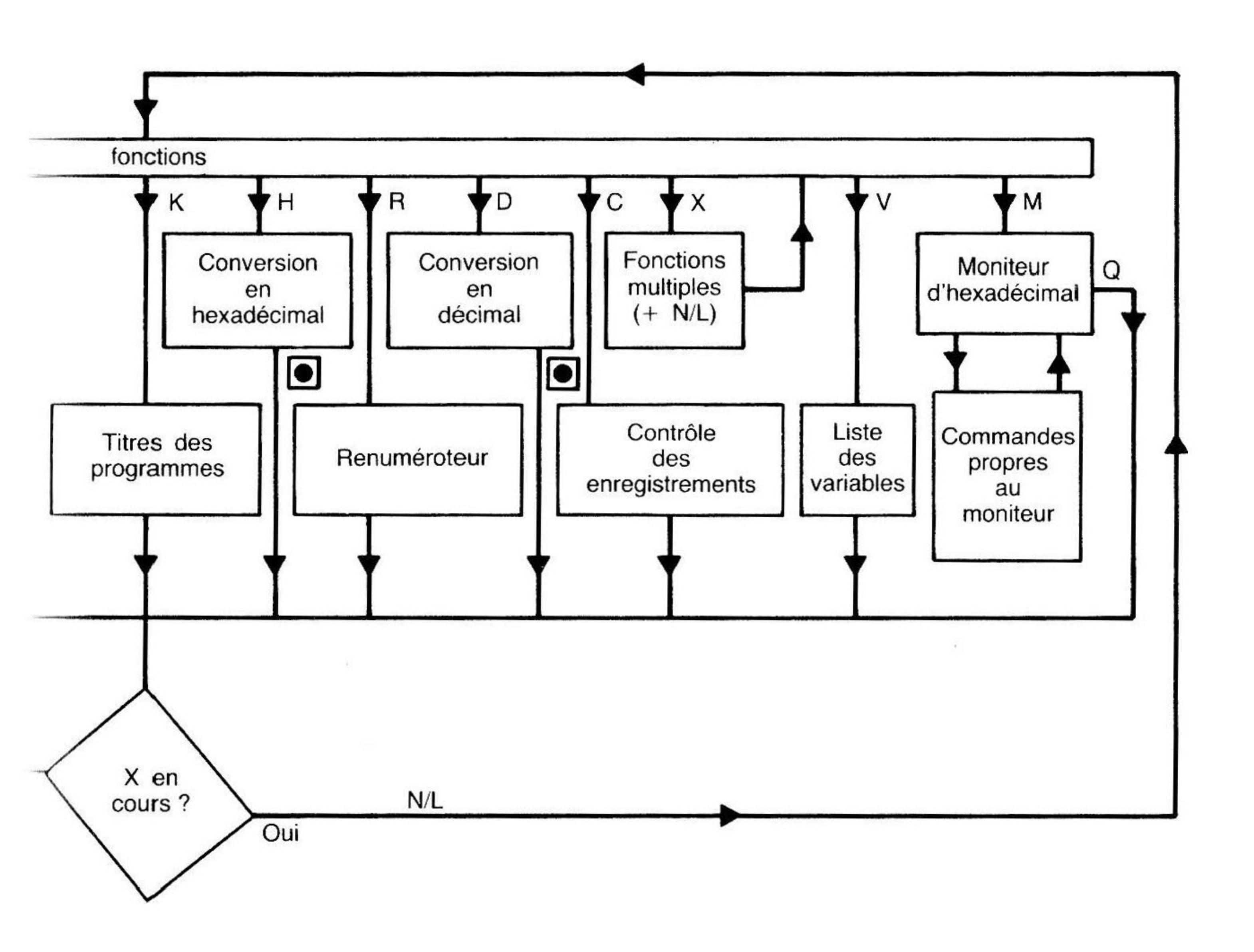
7FDD 77..........LD (HL),A
7FDE 23..........INC HL
7FDF C9..........RET
7FE0 23..........INC HL
7FE1 5E..........LD E,(HL)
7FE2 22.3C.40....LD (403C),HL
7FE5 23..........INC HL
7FE6 56..........LD D,(HL)
7FE7 23..........INC HL
7FE8 7E..........LD A,(HL)
7FE9 FE.E3.......CP E3
7FEB C9..........RET
7FEC 00..........NOP
7FED CD.B4.07....CALL 07B4
7FF0 20.08.......JR NZ,7FFA
7FF2 1B..........DEC DE
7FF3 1B..........DEC DE
7FF4 1B..........DEC DE
7FF5 1B..........DEC DE
7FF6 1B..........DEC DE
7FF7 FE.80.......CP 80
7FF9 C9..........RET

7FFA FE.DE.......CP DE
7FFC 1B..........DEC DE
7FFD 23..........INC HL
7FFE C9..........RET
```



UTILITAIRES - MODULES COMMUNS -

ENTRÉES AU CLAVIER - AFFICHAGE DU CONTENU DES REGISTRES - ETC...


fonctions
K
H
R
D
C
X
V
M
Conversion en hexadécimal
Conversion en décimal
Fonctions multiples (+ N/L)
Moniteur d'hexadécimal
Q
Titres des programmes
Renuméroteur
Contrôle des enregistrements
Liste des variables
Commandes propres au moniteur
X en cours ?
N/L
Oui

PILOTEZ VOTRE ZX-81

Dans cet ouvrage, l'auteur découvre avec vous le ZX-81 de SINCLAIR.
Des programmes originaux, qui peuvent être utilisés directement, sans notions de programmation, mettent en œuvre de nombreuses applications « domestiques » de l'informatique. L'étude progressive des instructions du BASIC et un tour complet des possibilités du ZX-81 dans sa version de base vous aideront à créer vos propres programmes.



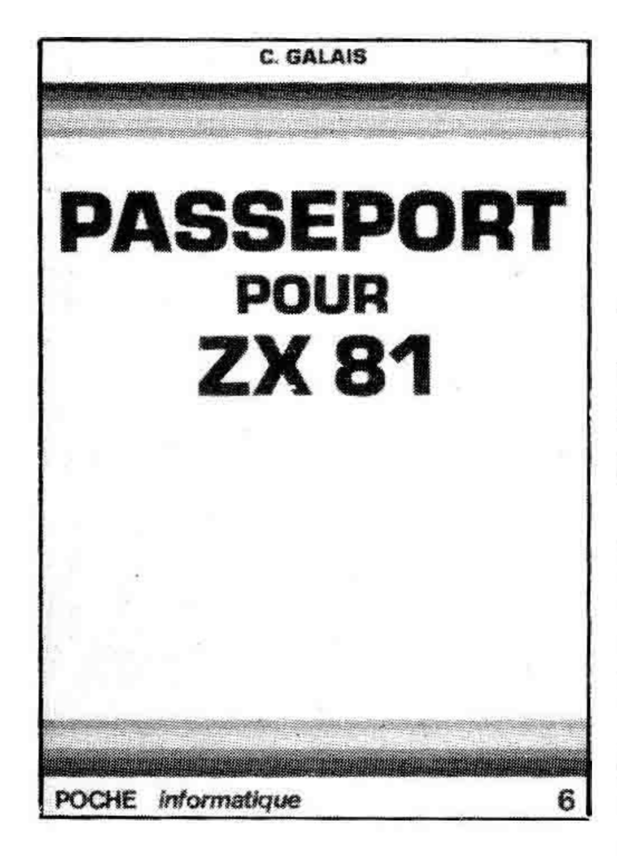
Principaux chapitres :

- **Prise de contact avec le ZX-81**
- **Jeux et divertissements**
- **Mathématiques**
- **Fonctions graphiques**
- **Fichiers et répertoires**
- **Annuaire électronique**
- **Ordinateur de bord automobile**
- **Echanges de programmes**

Les programmes qui figurent dans cet ouvrage ont été enregistrés sur cassette. Commandez-la à votre revendeur.

Un livre de 128 pages, format 15 × 21 cm

Edité par :



PASSEPORT POUR ZX 81

Un livre de 144 pages, format 11,7 × 16,5 cm

Toutes **les fonctions, instructions et commandes du ZX 81** sont présentées dans **l'ordre alphabétique.**
Leur recherche est donc facile et rapide.
Le **débutant** pourra s'initier à l'emploi de chaque mot clé grâce au **programme** et aux **explications** qui sont donnés pour chacun d'eux.
Pour celui qui maîtrise déjà le Basic du ZX 81, ce manuel sera un très utile **aide-mémoire** pour perfectionner sa programmation.

Méthode de présentation

Classement alphabétique des commandes, fonctions et instructions, avec pour chacune d'elles :

- la traduction anglais/français
- son utilisation
- un exemple de programme
- une explication détaillée

Edité par :

EDITIONS TECHNIQUES ET SCIENTIFIQUES FRANÇAISES
2 à 12, rue de Bellevue - 75940 PARIS CEDEX 19

Achevé d'imprimer
sur les presses
de l'Imprimerie Marcel Bon
70 Vesoul
Dépôt légal : avril 84
N° éditeur : 420 - N° imprimeur : 2773



http://zx81.ordi5.free.fr/

MICRO SYSTEMES



UTILITAIRES POUR ZX 81

Cet ouvrage vous fait découvrir le **langage machine du Z 80** et vous dévoile toutes les ressources matérielles et logicielles de votre système, jusqu'aux plus complexes comme le calculateur et les périphériques. **Des programmes utilitaires performants, écrits en assembleur,** pourront être employés directement. Ils serviront aussi d'exemples, grâce aux commentaires accompagnant chaque routine, pour créer de nouvelles applications, même sur d'autres machines.

Quelques applications :

- Désassembleur
- Aiguillage
- Transferts en REM
- Moniteur d'hexadécimal
- Générateur de REM longues
- Entrée d'une adresse en hexadécimal
- Sortie en mode SLOW
- Affichage des registres HL et A

ISBN 2-85535-070-0

TARIF CODE B


9
UTILITAIRES POUR ZX 81
M. SAAL